Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni nastavnički studij matematike i informatike

Marija Mecanović

Metoda sukcesivne nadrelaksacije (SOR)

Diplomski rad

Osijek, 2017.

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni nastavnički studij matematike i informatike

Marija Mecanović

Metoda sukcesivne nadrelaksacije (SOR)

Diplomski rad

Mentor: doc. dr. sc. Ivana Kuzmanović
Komentor: izv. prof. dr. sc. Zoran Tomljanović

Osijek, 2017.

Sadržaj

Uvod

Predmet ovog diplomskog rada je predstaviti metodu sukcesivne nadrelaksacije, skraćeno SOR metodu (engl. *Successive Overrelaxation Method*) kao metodu s bržom konvergencijom i većom točnošću od ostalih standardnih, najčešće korištenih, iterativnih metoda, te također prikazati njezinu učinkovitu implementaciju u programskom jeziku MATLAB.
Velik broj znanstvenika, te inženjera diljem svijeta koriste MATLAB za analizu i dizajniranje sustava i proizvoda mijenjajući naš svijet. Njegova platforma je optimizirana za rješavanje velikog broja znanstvenih i tehničkih problema, te je njegov programski jezik, koji je matrično orijentiran, najprirodniji način izražavanja računalne matematike. Ugrađena grafika olakšava vizualizaciju i dobivanje uvida iz podataka. Upravo iz tog razloga, MATLAB je pogodan alat za implementaciju i testiranje SOR metode.
Sustavi linearnih jednadžbi su prožeti kroz velik broj problema modeliranja. Svi smo upoznati s direktnim metodama za rješavanje sustava $Ax = b$, te smo se u velikoj većini susreli s tim problemom još u sedmom razredu osnovne škole, iako nam nije bio predstavljen u matričnom obliku. U slučajevima kada je broj jednadžbi i nepoznanica malen, ovaj problem je rješiv vrlo brzo i s velikom točnošću. Međutim, za velike sustave linearnih jednadžbi u kojima imamo velik broj nepoznanica i uvjeta, dolazi do znatnih problema. Čak i uz pomoć današnjih tehnološki vrlo razvijenih računala, ovaj problem može postati vrlo težak. U tu svrhu, došlo je do potrebe za razvijanjem učinkovitih metoda za rješavanje ovakvih velikih sustava. Te metode dijelimo u dvije osnovne skupine, direktne i iterativne metode. Tema ovog diplomskog rada je metoda sukcesivne nadrelaksacije, koja spada u iterativne metode, te će u ovom radu biti posvećena puno veća pažnja iterativnim metodama. Kao što samo ime kaže, pomoću iterativnih metoda želimo poboljšati početnu aproksimaciju, tako da u svakom koraku iteracije, greška u izračunu aproksimacije bude što manja. Nakon određenog broja iteracija, kada je zadovoljen kriterij zaustavljanja, odnosno kada dođemo do greške koju možemo tolerirati do na neku decimalu, dobit ćemo vektor koji ćemo smatrati dovoljno dobrom aproksimacijom rješenja. Međutim, isto tako se postavlja i pitanje čemu to? Zašto se zadovoljiti s "dovoljno dobrom aproksimacijom" ako uz pomoć direktnih metoda, kao što su Gaussove eliminacije, možemo doći do egzaktnog rješenja? Vrlo dobro je poznato da rješavanje linearnog sustava pomoću Gaussovih eliminacija zahtijeva zalihu u memoriji za skladištenje svih ulaza matrice A, odnosno n^2 elemenata, kao i otprilike $\frac{2}{3}n^3$ aritmetičkih operacija, zbrajanja, oduzimanja, množenja te dijeljenja. I upravo ta složenost u izračunavanju egzaktnog rješenja dovela je do razvoja velikog broja iterativnih metoda koje koriste samo množenje vektora i matrica, uz mali broj dodatnih operacija. Također, razvitku iterativnih metoda pridonijela je i spoznaja da matrice koje se pojavljuju u ovakvim problemima imaju i posebna svojstva. Često su matrice rijetko popunjenje, tj. velik broj elemenata im je jednak nuli. Rijetka popunjenost sustava i ostala posebna svojstva matrice mogu često biti korištena u svrhu velike prednosti pri matrično-vektorskom množenju. Ako matrica ima samo nekoliko ne-nul elemenata po retku, tada je broj operacija potreban za izračun produkta te matrice sa danim vektorom samo nekoliko n-ova, umjesto inače potrebnih $2n^2$ operacija pri gustom matričnom množenju. Prostor potreban za pohranu je potreban samo za ne-nul elemente matrice, te ako su i oni dovoljno jednostavni za izračunavanje, čak i to može biti izbjegnuto.

U prvom poglavlju ovog diplomskog rada bit će prestavljeni osnovni rezultati linearne algebre, odnosno osnovni pojmovi potrebni za lakše razumijevanje rezultata koji se nalaze u ovom diplomskom radu. Bit će definiran pojam matrice, dijagonalne matrice, trokutaste matrice, transponirane matrice, inverza matrice, te regularne matrice.

Direktne metode će biti predstavljene samo u svom idejnom smislu, te neće biti detaljnije analize algoritama direktnih metoda. Dani su i rezultati o složenosti direktnih metoda u svrhu usporedbe sa složenosti iterativnih metoda.
Nakon motivacije za uvođenje iterativnih metoda, bit će predstavljene dvije najpoznatije iterativne metode, Jacobijeva i Gauss-Seidelova, te će posebno, u drugom poglavlju, biti predstavljena SOR metoda, kao glavna tema ovog diplomskog rada. Također će se kroz rad isticati ujedno i složenost predstavljenih metoda kako bi bila dana što detaljnija analiza učinkovitosti pojedine metode. Ono što je najvažnije za ove tri metode jeste njihova konvergencija, pa će biti uspoređene njihove konvergencije, te prikazani rezultati na različitim primjerima.
U poglavlju koje se bavi SOR metodom i njenim opisom bit će predstavljena sama ideja metode i potreba za njenim nastankom, te će biti uveden pojam relaksacijskog parametra, tj. nadrelaksacijskog i podrelaksacijskog parametra čiji je izbor vrlo važan u implementaciji SOR metode. Bit će predstavljeni i rezultati o konvergenciji SOR metode koji će biti primijenjeni na primjerima u posljednjem poglavlju.
Na kraju, u posljednjem poglavlju bit će sumirani svi kriteriji konvergencije pojedinih metoda, te će biti uspoređene brzine konvergencije i točnosti Jacobijeve, Gauss-Seidelove te SOR metode s različitim relaksacijskim parametrima i to za nekoliko klasa problema gdje će biti donešen i konačan zaključak o navedenim metodama, te njihovim konvergencijama. Također su priloženi i objašnjeni MATLAB kodovi Jacobijeve, Gauss-Seidelove, te SOR metode u programskom jeziku MATLAB, te grafički prikazi ovisnosti greške o iteracijama dobiveni primjenom iterativnih metoda u MATLAB-u na primjerima iz posljednjeg poglavlja.

U dodatku ovom diplomskom radu priložen je MATLAB kod za grafičko korisničko sučelje pomoću kojega su dobiveni grafički prikazi ovisnosti greške o iteracijama Jacobijeve, Gauss-Seidelove i SOR metode.

1 Sustavi linearnih jednadžbi

Matematika se počela razvijati prije više tisuća godina, još u doba starih Egipćana. Razvila se iz potrebe za obavljanjem proračuna u trgovini, mjerenjem zemljišta i predviđanjem astronomskih događaja, a iza svake od te tri primjene matematike stajao je određeni problem s kojim su se tadašnji ljudi susretali. Zapravo, uzrok razvoja svake pojedine grane matematike je upravo postavljanje problema, te njegovo rješavanje, pa je tako i proučavanje linearne algebre a samim time i numeričke linearne algebre, započelo postavljanjem problema. U ovom slučaju on glasi: *"Pomoću elementarnih aritmetičkih operacija naći sve n-torke realnih brojeva $x_1, x_2, \ldots, x_n$ koje istovremeno zadovoljavaju m jednadžbi."* Javlja se potreba za što elegantnijim rješavanjem ovakvog problema, te uslijed toga dolazi do ideje primjene matrica. U svrhu što boljeg shvaćanja teorije koja leži u pozadini ovog problema, ponovit ćemo osnovne pojmove vezane uz matrice.

Definicija 1.1 *Neka su m i n prirodni brojevi. Svaku familiju A realnih brojeva a_{ij} pri čemu je $1 \leq i \leq m$, te $1 \leq j \leq n$, zapisujemo u obliku pravokutne tablice*

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

i zovemo ***matrica*** *s* ***m redaka*** *i* ***n stupaca***. *Element a_{ij} se nalazi na mjestu (i, j) u matrici A.*
Dvije matrice $A = (a_{ij})$ i $B = (b_{ij})$ su ***jednake*** *ako su istih dimenzija, odnosno imaju jednak broj stupaca i jednak broj redaka, te vrijedi da je $a_{ij} = b_{ij}$, $\forall i, j$, $1 \leq i \leq m$, $1 \leq j \leq n$.*
Za matricu A kažemo da je ***kvadratna matrica n-tog reda*** *ako je $m = n$.*
Kvadratna matrica A je ***dijagonalna*** *ako je oblika*

$$A = \begin{bmatrix} a_{11} & 0 & \ldots & 0 \\ 0 & a_{22} & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & a_{nn} \end{bmatrix}$$

Elementi $a_{11}, a_{22}, \ldots, a_{nn}$ čine ***glavnu dijagonalu*** *matrice A.*
Kvadratna matrica A je ***gornje trokutasta matrica*** *ako su joj svi elementi ispod glavne dijagonale jednaki nula, odnosno $a_{ij} = 0$ za $i > j$.*
Kvadratna matrica je ***donje trokutasta*** *ako su joj svi elementi iznad glavne dijagonale jednaki nula, odnosno $a_{ij} = 0$ za $i < j$.*
Gornje i donje trokutaste matrice zajednički nazivamo ***trokutaste matrice***.
Veliki značaj u proučavanju matrica imaju sljedeće kvadratne matrice

$$I = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & 1 \end{bmatrix}, \qquad O = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & 0 \end{bmatrix}.$$

Matricu I zovemo ***jedinična matrica***, *a O* ***nul-matrica*** *n-tog reda.*

Definicija 1.2 ***Transponirana matrica*** *matrice $A = (a_{ij})$ je matrica $B = (b_{ji})$ koja ima n redaka i m stupaca, te za koju je $b_{ji} = a_{ij}$, pri čemu je $1 \leq i \leq m$, $1 \leq j \leq n$.*
Transponiranu matricu matrice A označavamo s A^T.

Definicija 1.3 *Matrica $A \in \mathbb{R}^{n\times n}$ je* ***simetrična*** *ako je jednaka svojoj transponiranoj matrici, $A = A^T$, tj. $a_{ij} = a_{ji}$, $\forall i, j$ za koje vrijedi $1 \leq i, j \leq n$.*

Definicija 1.4 *Simetrična matrica $A \in \mathbb{R}^{n\times n}$ je* ***pozitivno definitna*** *ako za svaki vektor $x \in \mathbb{R}^n \setminus \{0\}$ vrijedi da je $x^T Ax > 0$.*
Također, simetrična matrica $A \in \mathbb{R}^{n\times n}$ je pozitivno definitna ako i samo ako su sve svojstvene vrijednosti od A strogo veće od 0. (Za dokaz vidi [4].)

Definicija 1.5 *Neka je matrica A iz skupa svih kvadratnih matrica $\mathcal{M}_n$. Matrica $B \in \mathcal{M}_n$ sa svojstvom*

$$AB = BA = I$$

zove se ***inverzna matrica*** *matrice A i piše se $B = A^{-1}$.*
Kvadratna matrica koja ima inverznu matricu zove se ***regularna*** *(invertibilna, nesingularna) matrica. Matrica je* ***singularna*** *ako nije regularna.*

Vratimo se na početni problem od m jednadžbi s n nepoznanica. Pod pojmom sustava linearnih jednadžbi podrazumijevamo skup jednadžbi sljedećeg oblika:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1, \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2, \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & = & b_m \end{array} \tag{1.1}$$

pri čemu skalare a_{ij} nazivamo koeficijentima jednadžbe, skalare b_i slobodnim članovima, a x_j nepoznanicama, za $1 \leq i \leq m$ i $1 \leq j \leq n$. Očigledno se ovaj sustav (1.1) može zapisati na sljedeći način:

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \qquad 1 \leq i \leq m \tag{1.2}$$

što nas motivira za uvođenje matričnog zapisa početnog sustava m jednadžbi s n nepoznanica.

Definicija 1.6 *Neka je $A \in \mathbb{R}^{m\times n}$ matrica dana s*

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \tag{1.3}$$

Neka je vektor nepoznanica vektor $x \in \mathbb{R}^n$ oblika x_j pri čemu je $1 \leq j \leq n$, a za vektor $b \in \mathbb{R}^m$ uzmimo da je vektor slobodnih članova b_i za $1 \leq i \leq m$. Uzimajući u obzir jednakost (1.2) i definiciju matričnog množenja, sustav od m linearnih jednadžbi s n nepoznanica možemo zapisati u obliku

$$Ax = b \tag{1.4}$$

pri čemu je A oblika (1.3) matrica sustava, x vektor rješenja, a b vektor desne strane.

Sve bitne informacije o sustavu, nose upravo ovi elementi. U svrhu lakšeg promatranja ovog sustava, možemo ga zapisati i u matrično - vektorskoj notaciji:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Naš je zadatak pronaći vektor $x = [x_1 \quad x_2 \quad \dots \quad x_n]^T$ koji zadovoljava ovaj sustav jednadžbi. Njega ćemo pronaći pomoću određenih metoda za rješavanje sustava linearnih jednadžbi koje uglavnom dijelimo u dvije osnovne skupine: direktne i iterativne. Direktne metode izračunavaju potpuno točan, egzaktan, rezultat u konačnom broju koraka za bilo koji sustav linearnih jednadžbi koji ima rješenje. Međutim, takav pristup rješavanju sustava ima ozbiljan nedostatak u pogledu vremena i prostora potrebnog za rješavanje složenijih sustava. S druge strane, iterativne metode u pravilu ne mogu izračunati u potpunosti točan rezultat u konačnom broju koraka, već su zamišljene tako da počinju s aproksimacijom rješenja i svakim svojim korakom smanjuju odstupanje, tj. udaljenost od točnog rješenja, odnosno pogrešku u aproksimaciji rješenja. Sve to ima smisla naravno, samo ukoliko iterativni postupak konvergira. Cijeli postupak se zaustavlja onda kada je pogreška aproksimacije rješenja manja od neke dozvoljene tolerancije, tj. prihvatljive vrijednosti do na koju možemo tolerirati odstupanje. Kako bi iterativna metoda bila učinkovita, krucijalno je postojanje dobrih kriterija za određivanje trenutka kada treba stati s iteracijama, odnosno kada je trenutna aproksimacija rješenja dovoljno blizu točnom rješenju. Karakteristike dobrih kriterija za zaustavljanje iterativnog procesa uključuju određivanje kada je razlika između $x^{(k)}$ i $x^{(k+1)}$ aproksimacija dovoljno mala, zaustavljanje iterativnog procesa ukoliko se pogreška smanjuje previše sporo, te ukoliko se uoči divergencija metode. Tri su najvažnija kriterija zaustavljanja iterativnih procesa u metodama za određivanje rješenja linearnog sustava (1.4): maksimalan broj koraka iteracije, dozvoljena norma reziduala $||Ax - b||$ treba biti manja od zadane tolerancije, te iterativna pogreška nakon k iteracija definirana s $||x^{(k+1)} - x^{(k)}||$.

Nedostatak iterativnih metoda je u tome što se ne mogu primijeniti na bilo koji problem koji možemo izraziti sustavom linearnih jednadžbi. Postoje sustavi linearnih jednadžbi za koje se iterativne metode neće niti približavati točnom rješenju, tj. za te sustave će divergirati.

U sljedeća dva potpoglavlja bit će detaljnije objašnjene, te navedene neke od direktnih, odnosno iterativnih metoda.

1.1 Direktne metode za rješavanje linearnih sustava

Idejna struktura direktnih metoda se zasniva na različitim faktorizacijama matrice sustava koje omogućavaju jednostavno rješavanje sustava. Dakle, ako se matrica A prikaže pomoću produkta jednostavnijih matrica M i N, tj. $A = MN$, onda sustav $Ax = b$ prelazi u sustav $MNx = b$.
Zatim se, uzimajući supstituciju $Nx = y$ rješava sustav $My = b$, kao jednostavniji linearni sustav, a nakon toga sustav $Nx = y$. Na taj način se dobiva vektor x koji rješava naš početni sustav $Ax = b$.
Osnovna metoda koja pripada grupi direktnih metoda je metoda Gaussovih eliminacija koja

vrši faktorizaciju matrice sustava na produkt trokutastih matrica, takozvanu LU faktorizaciju matrice.

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}.$$

Pri tome je matrica L donje trokutasta matrica s jedinicama na dijagonali, a matrica U gornje trokutasta regularna matrica, tj. svi dijagonalni elementi su joj različiti od 0.
Ukoliko vrijedi da su sve glavne podmatrice A_j, matrice A, regularne, tada postoji jedinstvena LU faktorizacija matrice A i tada je rješavanje sustava vrlo jednostavno. Sustav $Ax = b$ zamijenjujemo sustavom $LUx = b$, te se on može zapisati kao

$$Ly = b \tag{1.5}$$

$$Ux = y \tag{1.6}$$

i prema tome se svodi na dva sustava s trokutastim matricama. Još preostaje konstruirati metode za rješavanje tih sustava. Sustav (1.5) se rješava pomoću algoritma supstitucija unaprijed, a potom se rješava sustav (1.6) pomoću algoritma povratnih supstitucija.
Pogledajmo sada složenost ovakve direktne metode kroz dvije propozicije koje neće biti dokazane. Za dokaz pogledati [8] (str. 66., 67., Lema 5.2, Lema 5.3).

Propozicija 1.1.1 *Složenost računanja algoritma za LU faktorizaciju kvadratne matrice reda n iznosi:*

$$C(n) = \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n \tag{1.7}$$

Propozicija 1.1.2 *Računska složenost supstitucija unaprijed i supstitucija unazad iznosi*

$$C(n) \sim n^2 \tag{1.8}$$

Iz (1.7) i (1.8) lako se da zaključiti da je računska složenost Gaussovih eliminacija reda veličine

$$C(n) \sim \frac{2}{3}n^3.$$

Još jedna vrlo bitna faktorizacija matrice sustava je QR faktorizacija, odnosno dekompozicija. To je rastav matrice $A \in \mathbb{R}^{m\times n}$ na produkt dviju matrica Q i R, tj. $A = QR$ pri čemu je $Q \in \mathbb{R}^{m\times m}$ unitarna matrica, a $R \in \mathbb{R}^{m\times n}$ gornje trokutasta. Pri ovoj faktorizaciji ne zahtijeva se da je matrica sustava kvadratna, samo da je $m \geq n$, odnosno da je broj redaka veći ili jednak broju stupaca.
Neke od ostalih direktnih metoda su Cramerovo pravilo, LR faktorizacija, Gauss-Jordanova metoda, Thomasov algoritam, itd.

Kako tema ovog diplomskog rada nisu direktne metode, već iterativne, neće biti predstavljena detaljnija analiza algoritma za Gaussove eliminacije, niti za QR dekompoziciju, već se ovim potpoglavljem samo htjelo osvrnuti na ideju direktnih metoda te dobiti osjećaj vremenske složenosti ovakvih metoda.

1.2 Iterativne metode

Umjesto direktnih metoda za rješavanje sustava, u praksi se vrlo često koriste iterativne metode jer su puno učinkovitije za velike rijetko popunjene sustave (engl. *sparse*), tj. sustave koji imaju vrlo velik broj elemenata jednakih nuli.
Neka je dana matrica $A \in \mathbb{R}^{n\times n}$ i neka su M,$N \in \mathbb{R}^{n\times n}$ takve da je

$$A = M + N \tag{1.9}$$

pri čemu je matrica M izabrana tako da se njen inverz računa lakše od inverza matrice A. Tada dani sustav $Ax = b$ možemo zapisati u obliku

$$Mx = -Nx + b$$

pri čemu je M regularna matrica. Ako nam je dana aproksimacija $x^{(k)}$ tada možemo izračunati $(k+1)-vu$ iteraciju tj. $(k+1)-vu$ aproksimaciju rješenja kao rješenje jednadžbe

$$Mx^{(k+1)} = b - Nx^{(k)} \qquad k = 0, 1, 2, ... \tag{1.10}$$

Pretpostavimo li da vrijedi $\lim\limits_{k\to\infty} x^{(k+1)} = x$, tada za limes x vrijedi

$$Mx = \lim_{k\to\infty} Mx^{(k+1)} = \lim_{k\to\infty} (b - Nx^{(k)}) = b - Nx.$$

Lako se uoči da za taj limes vrijedi $Ax = b$. To nam pokazuje da u slučaju konvergencije niza (1.10) njegov limes predstavlja rješenje sustava linearnih jednadžbi.
U svrhu proučavanja konvergencije iterativne metode, proučavamo ponašanje pogreške $e^{(k+1)} = x^{(k+1)} - x$ gdje je x točno, odnosno egzaktno rješenje sustava linearnih jednadžbi. Tada slijedi:

$$Me^{(k+1)} = Mx^{(k+1)} - Mx = (b - Nx^{(k)}) - (A - N)x = -Ne^{(k)}$$

odnosno $e^{(k+1)} = -M^{-1}Ne^{(k)}$, za sve $k \in \mathbb{N}$. Prema tome, promatrana iterativna metoda će konvergirati ako i samo ako $e^{(k+1)} \to 0$ za $k + 1 \to \infty$, tj. $e^{(k)} \to 0$ za $k \to \infty$. Sljedeći teorem karakterizira konvergenciju iterativne metode pomoću svojstava spektralnog radijusa matrice

$$C = -M^{-1}N \tag{1.11}$$

pa ćemo se u svrhu toga prisjetiti definicije spektralnog radijusa matrice.

Definicija 1.2.1 *Spektralni radijus matrice $A \in \mathbb{R}^{n\times n}$ definiramo sa*

$$\rho(A) = max\ \{|\lambda| : \lambda\ svojstvena\ vrijednost\ matrice\ A\}$$

Teorem 1.2.1 *Neka je dana matrica $C \in \mathbb{R}^{n\times n}$ i vektori $e^{(0)} \in \mathbb{R}^n$ i $e^{(k)} = C^k e^{(0)} \in \mathbb{R}^n$, za sve $k \in \mathbb{N}$. Tada $e^{(k)} \to 0$ za svaki $e^{(0)} \in \mathbb{R}^n$ ako i samo ako je $\rho(C) < 1$.*

Dokaz. Pretpostavimo da je $\rho(C) < 1$. Tada možemo pronaći induciranu matričnu normu takvu da je $||C|| < 1$ pa imamo

$$||e^{(k)}|| = ||C^k e^{(0)}|| \leq ||C||^k ||e^{(0)}|| \to 0$$

za $k \to \infty$. (Vidi [8])
No međutim, ako je $\rho(C) \geq 1$, tada postoji barem jedan $e^{(0)} \in \mathbb{R}^n$ za koji vrijedi da je $Ce^{(0)} = \lambda e^{(0)}$ za neki $\lambda \in \mathbb{R}$ takav da je $|\lambda| \geq 1$. Za taj vektor $e^{(0)}$ dobivamo

$$||e^{(k)}|| = ||C^k e^{(0)}|| = |\lambda|^k ||e^{(0)}||$$

iz čega se vidi da $e^{(k)}$ ne konvergira ka 0 za $k \to \infty$. □

Napomena 1.2.1 *Posljedica gornjeg teorema je ocjena konvergencije iterativnog postupka (1.10). Postupak će konvergirati ako i samo ako je $\rho(C) < 1$. Brzina konvergencije je veća što je $\rho(C)$ manji, stoga slijedi ocjena:*

$$||x^{(k+1)} - x^{(k)}|| \leq \rho(C)||x^{(k)} - x|| \tag{1.12}$$

gdje je $||\cdot||$ bilo koja matrična norma.

Nakon što smo u prethodnom teoremu naveli uvjete konvergencije iterativnih metoda, pozabavit ćemo se još malo samim iterativnim metodama.
Iterativne metode mogu se podijeliti u dvije osnovne skupine: stacionarne iterativne metode i nestacionarne iterativne metode. Stacionarne iterativne metode su starije i jednostavnije za razumijevanje, ali su obično manje učinkovite od nestacionarnih iterativnih metoda. U skupinu stacionarnih metoda spadaju sljedeće metode: Jacobijeva metoda, Gauss-Seidelova metoda, SOR metoda, itd.
Nestacionarne iterativne metode su razvijene relativno nedavno. Njihova analiza je obično teža za razumijevanje, pa su time i teže za implementaciju, no njihova učinkovitost može biti izuzetno velika. Ova skupina metoda razlikuje se od stacionarnih metoda po tome što njihovi izračuni uključuju informacije koje se mijenjaju sa svakim korakom iteracije. Tipično je da se konstante izračunavaju skalarnim umnošcima vektora ostataka i drugih vektora koji nastaju samom iterativnom metodom. U tu skupinu spadaju sljedeće metode: metoda konjugiranih gradijenta (engl. *Conjugate Gradient Method*), metoda minimalnog ostatka (engl. *Minimum Residual Method*), metoda konjugiranih gradijenta na normalnim jednadžbama (engl. *Conjugate Gradient on the Normal Equations Method*), metoda dvostrukog konjugiranog gradijenta (engl. *BiConjugate Gradient Method*), metoda kvadratnog konjugiranog gradijenta (engl. *Conjugate Gradient Squared Method*), Chebyshevljeva iteracija (engl. *Chebyshev Iteration*), itd.

U sljedeća dva potpoglavlja bit će dan detaljniji pregled dvije osnovne stacionarne metode.

1.2.1 Jacobijeva metoda

Pogledajmo sljedeći rastav matrice $A \in \mathbb{R}^{n \times n}$

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ a_{21} & 0 & \dots & 0 & 0 \\ a_{31} & a_{32} & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn-1} & 0 \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \dots & 0 & a_{nn} \end{bmatrix},$$

$$\mathbf{U} = \begin{bmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & 0 & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1n} \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}. \tag{1.13}$$

Jasno se vidi da je $A = L + D + U$.

Iterativna metoda koju dobivamo iz (1.9) za izbor matrica $M = D$ i $N = L + U$, gdje su L, D i U definirane kao u (1.13) zovemo *Jacobijeva metoda*. S obzirom na izbor matrica M i N, iterativni postupak poprima sljedeći oblik:

$$x^{(k+1)} = D^{-1}(b - (L + U)x^{(k)})$$

za sve $k \in \mathbb{N}$, pa konvergencija promatrane metode ovisi o svojstvima matrice C iz (1.11), koja za ovu metodu ima oblik $C_J = -D^{-1}(L + U)$. Uzevši u obzir da je matrica D dijagonalna, njezin inverz se izračunava trivijalno, pa time konstrukcija matrice C_J nije *skupa*[1], vremenska složenost je $O(n)$.
Ono što nas osobito zanima kod ove iterativne metode je njezina konvergencija. No, međutim, samo za matrice s posebnim svojstvima postoje rezultati koji govore o tome. Stoga najprije definirajmo pojam *strogo dijagonalne dominantnosti*.

Definicija 1.2.1.1 *Za matricu A kažemo da je:*

(a) strogo dijagonalno dominantna po recima ako vrijedi

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \qquad i = 1, \dots, n$$

(b) strogo dijagonalno dominantna po stupcima ako vrijedi

$$|a_{jj}| > \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}|, \qquad j = 1, \dots, n$$

Za matrice koje zadovoljavaju takve uvjete vrijedi sljedeći teorem.

Teorem 1.2.1.1 *Neka je matrica A strogo dijagonalno dominantna i po recima i po stupcima. Tada Jacobijeva metoda konvergira.*

Dokaz. Lako se vidi da su elementi matrice $C_J = -D^{-1}(L + U)$ oblika

$$c_{ij} = \begin{cases} -\frac{a_{ij}}{a_{ii}}, & \text{ako je } i \neq j \\ 0, & \text{za } i = j \end{cases}$$

[1]Nije potrebno puno računskih operacija za njezin izračun.

Tada prema Definiciji 2.1.1 vrijedi

$$||C_J||_\infty = \max_{i=1,\dots,n} \sum_{j=1}^{n} |(C_J)_{ij}| = \max_{i=1,\dots,n} \frac{1}{|a_{ii}|} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|$$

Stroga dijagonalna dominacija matrice povlači da je $||C_J||_\infty < 1$, tj. da je $\rho(A) < 1$, a to prema Teoremu 1.2.1 znači da metoda konvergira. □

1.2.2 Gauss-Seidelova metoda

U Jacobijevoj metodi smo mogli primjetiti jedan nedostatak, a to je da se pri izračunavanju $(k+1)$-ve iteracije svaka komponenta vektora $x^{(k+1)}$ izračunavala pomoću komponenata vektora $x^{(k)}$, tj. računali smo ih sekvencionalno, odnosno serijski. Međutim, ako bismo Jacobijevu metodu modificirali na način da prethodno izračunate komponente aproksimacije $x^{(k+1)}$ koristimo prilikom računanja sljedećih komponenti tog istog vektora, omogućili bismo da komponente od $x^{(k+1)}$ prebrišu stare vrijednosti komponenti od $x^{(k)}$ čim se izračunaju, dok su u Jacobijevoj metodi ostajale. Onda nam više ne bi bio potreban dodatni pomoćni vektor za pamćenje komponenti prethodne iteracije $x^{(k)}$. Opisanu prednost nalazimo u Gauss-Seidelovoj metodi.
Za matrice M i N iz (1.9) uzmimo $M = L + D$, a $N = U$. Tada imamo sljedeću formulu za iterativni postupak:

$$(L + D)x^{(k+1)} = b - Ux^{(k)}$$

pri čemu je matrica konvergencije za Gauss-Seidelovu metodu $C_{GS} = -(L + D)^{-1}U$
Prema tome, komponente vektora $x^{(k+1)}$ izračunavaju se na sljedeći način:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \qquad i = 1, \dots, n$$

S obzirom na navedene razlike između Jacobijeve i Gauss-Seidelove metode i uzevši činjenicu da je Gauss-Seidelova metoda poboljšanje Jacobijeve, moglo bi se pomisliti da ona brže konvergira od Jacobijeve, no to nije uvijek slučaj.
U sljedećem poglavlju će na primjerima biti pokazni detaljniji rezultati konvergencije ove dvije metode.

1.2.3 Neki rezultati o konvergenciji

Primjer 1.2.3.1 *Za danu matricu A ispitajmo konvergiraju li Jacobijeva i Gauss-Seidelova metoda.*

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Matrice konvergencije izgledaju ovako:

$$C_J = \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad C_{GS} = \frac{1}{16} \begin{bmatrix} 0 & 8 & 0 & 0 \\ 0 & 4 & 8 & 0 \\ 0 & 2 & 4 & 8 \\ 0 & 1 & 2 & 4 \end{bmatrix}.$$

Iz toga slijedi:

$$\rho(C_J) = 0.809, \qquad \rho(C_{GS}) = 0.6545.$$

Jasno se vidi da obje metode konvergiraju, ali da Gauss-Seidelova metoda konvergira brže.

U sljedeća dva primjera ćemo pogledati situacije kada jedna od metoda konvergira a druga ne.

Primjer 1.2.3.2 *Provjerimo konvergenciju Jacobijeve i Gauss-Seidelove metode na matrici*

$$A = \begin{bmatrix} 1 & -2 & 2 \\ -1 & 1 & -1 \\ -2 & -2 & 1 \end{bmatrix}$$

Za matricu A matrice konvergencije izgledaju ovako:

$$C_J = \begin{bmatrix} 0 & 2 & -2 \\ 1 & 0 & 1 \\ 2 & 2 & 0 \end{bmatrix}, \qquad C_{GS} = \begin{bmatrix} 0 & 2 & -2 \\ 0 & 2 & -1 \\ 0 & 8 & -6 \end{bmatrix}.$$

Slijedi

$$\rho(C_J) = 0.1064 \cdot 10^{-4}, \qquad \rho(C_{GS}) = 2(1 + \sqrt{2}),$$

što znači da Jacobijeva metoda konvergira, a Gauss-Seidelova divergira.

Primjer 1.2.3.3 *Neka je dana matrica*

$$A = \frac{1}{2} \begin{bmatrix} 2 & 1 & -1 \\ -2 & 2 & -2 \\ -1 & 1 & 2 \end{bmatrix}.$$

Tada za matrice konvergencije C_J i C_{GS} vrijedi da je

$$\rho(C_J) = 1, \qquad \rho(C_{GS}) = \frac{1}{2},$$

pa se sada lako vidi da Gauss-Seidelova metoda konvergira a Jacobijeva ne.

Kao zaključak ćemo navesti nekoliko teorema o konvergenciji ove dvije metode, ali ih nećemo dokazivati. (Za dokaz vidi [3].)

Teorem 1.2.3.1 *Ako je matrica $A \in \mathbb{R}^{n\times n}$ simetrična i pozitivno definitna, onda Gauss-Seidelova metoda za rješavanje sustava $Ax = b$ konvergira za svaki početni vektor $x^{(0)}$.*

Teorem 1.2.3.2 *Neka je A trodijagonalna matrica*[2]*. Tada ili i Jacobijeva i Gauss-Seidelova metoda za rješavanje sustava $Ax = b$ konvergiraju ili obje divergiraju. Ako obje konvergiraju, Gauss-Seidelova metoda konvergira brže.*

[2]Ima netrivijalnu glavnu dijagonalu, te dijagonalu iznad i ispod, dok su ostali elementi jednaki 0.

Teorem 1.2.3.3 *Ako je A strogo dijagonalno dominantna po recima, Jacobijeva i Gauss-Seidelova metoda konvergiraju. Preciznije,*

$$||C_{GS}||_\infty \leq ||C_J||_\infty < 1.$$

Unatoč navedenim primjerima i rezultatima da je pod nekim uvjetima Gauss-Seidelova metoda brža od Jacobijeve, ne postoji nikakav generalni rezultat te vrste. Čak štoviše, postoje nesimetrične matrice za koje Jacobijeva metoda konvergira, a Gauss-Seidelova ne, kao i za koje Gauss-Seidelova konvergira a Jacobijeva divergira, što smo mogli vidjeti iz Primjera 1.2.3.2 i Primjera 1.2.3.3.

2 SOR metoda

Nakon što shvatimo bit iterativnih metoda, te znamo konstruirati iterativni proces, nameće nam se ideja za uvođenjem jednog realnog parametra u cijeli proces, pa će radi lakšeg shvaćanja same SOR metode, prvo biti rečeno nešto više o njemu.

2.1 Relaksacijski parametar

Relaksacijski parametar se uvodi u iterativne metode s ciljem smanjenja spektralnog radijusa matrice konvergencije a samim time i ubrzanja konvergencije. Kako to postići? Zapravo, možemo nove aproksimacije u cijelom postupku računati u dva koraka. Prvo iz $x^{(k)}$ nađemo jednostavnu pomoćnu sljedeću aproksimaciju $x^{(k+1)*}$ a zatim za "pravu" novu aproksimaciju $x^{(k+1)}$ uzmemo težinsku sredinu prethodne aproksimacije $x^{(k)}$ i pomoćne nove aproksimacije $x^{(k+1)*}$

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega x^{(k+1)*} = x^{(k)} + \omega(x^{(k+1)*} - x^{(k)}),$$

gdje je ω težinski parametar kojeg možemo birati. Očito za $\omega = 1$ dobivamo $x^{(k+1)} = x^{(k+1)*}$, pa je ova metoda zapravo proširenje standardnih iterativnih metoda. Za ω je uobičajeno uzeti realan broj različit od 0, kako ne bismo dobili stacionaran niz[3].
Ideju korištenja parametra pronalazimo u općim metodama za rješavanje jednadžbi i minimizaciju funkcionala. Cijeli postupak je zapravo u svrhu približavanja pravom rješenju, odnosno smanjenja reziduala $r(x) = Ax - b$ u nekoj normi, tj. približavanja točki minimuma.

Naziv "relaksacija" dolazi upravo iz minimizacijskih metoda, a odnosi se na sve iterativne metode koje koriste neki oblik minimizacije reziduala. Zbog toga se često koristi i tradicionalni naziv "relaksacijski parametar" za ω. S obzirom na vrijednost parametra ω, imamo tri različita slučaja:

(a) $\omega = 1$, pri kojem se metoda svodi na pomoćnu metodu i to je tzv. obična ili standardna relaksacija

(b) $\omega < 1$, onda takvu metodu zovemo podrelaksacija (engl. *underrelaxation*)

(c) $\omega > 1$ onda metodu zovemo nad- ili pre-relaksacija (engl. *overrelaxation*)

One vrijednosti parametra ω za koje je $\rho(C)$ općenito najmanji za matricu konvergencije C iterativne metode, zovemo optimalnim relaksacijskim parametrima i označavamo s ω_{opt}. Srećom, za neke klase linearnih sustava, koje su izrazito bitne u primjeni, unaprijed možemo dobro odabrati ω_{opt} za maksimalno ubrzanje konvergencije iterativnih metoda i to tako da isti ω_{opt} vrijedi za sve iteracije. U većini slučajeva dobivamo $\omega_{opt} > 1$, pa se takve metode standardno zovu "OverRelaxation" i skraćeno označavaju s OR. S obzirom na to da se ω_{opt} zadaje ili bira unaprijed, a zatim koristi za sve iteracije, metoda je ovisna o jednom parametru i standardno koristimo oznaku OR(ω).

2.2 Opis SOR metode

David M. Young, Jr. (20.10.1923. - 21.12.2008) je američki matematičar i informatičar koji je jedan od začetnika moderne numeričke analize. On je 1950. u svojoj doktorskoj disertaciji začeo ideju metode sukcesivne nadrelaksacije (engl. *Successive Overrelaxation Method*),

[3]Niz kod kojeg su elementi međusobno jednaki.

skraćeno SOR(ω), ukazavši na prednosti dobivene u nekim slučajevima korištenjem fiksnog relaksacijskog parametra ω.
Motivacija mu je svakako bila poboljšanje Gauss-Seidelove petlje uzimajući odgovarajuću težinsku sredinu aproksimacija $x^{(k+1)}$ te $x^{(k)}$ što se odražava na poboljšanje svake pojedine varijable $x_j^{(k)}$, tj. pojedinačnih komponenti vektora $x^{(k)}$. Cilj takvog poboljšanja nije bio proširenje klase matrica za koje metoda konvergira, već ubrzanje konvergencije.
SOR je najčešće korištena stacionarna iterativna metoda. Kao što smo ranije spomenuli, u današnje vrijeme se češće koriste nestacionarne iterativne metode, upravo zbog razlike koju smo u poglavlju 1.2 objasnili, no stacionarne metode se uglavnom koriste za prekondicioniranje, tj. modificiranje originalnog linearnog sustava što za svrhu ima olakšavanje rješavanja danog linearnog sustava.
Za naš problem konvergencije, vrlo je bitno transformirati postojeći sustav u njemu ekvivalentan koji ima bolja spektralna svojstva (Teorem 1.2.1)

Proučimo samu ideju SOR metode. Za početak, kao i uvijek, promatramo sustav $Ax = b$, za $A \in \mathbb{R}^{n\times n}$ uz jednu pretpostavku, a to je da su dijagonalni elementi matrice ne-nul elementi, tj. $a_{ii} \neq 0$, $\forall i = 1, \ldots, n$.
Zatim, kao i kod Jacobijeve i Gauss-Seidelove metode, rastavimo matricu A na zbroj dviju matrica M i N, a za njihov izbor uzmimo:

$$M = \frac{1}{\omega}D + L \qquad N = \frac{\omega - 1}{\omega}D + U$$

gdje su L, D i U matrice definirane kao u (1.13).
Vidimo da se ovdje pojavio ω, relaksacijski parametar, tj. realan broj različit od nule.
Sada, uvrštavanjem u početni problem $Ax = b$ dobijemo

$$\begin{aligned} (M+N)x &= b \\ Mx + Nx &= b \\ Mx &= -Nx + b \end{aligned}$$

odnosno

$$\left(\frac{1}{\omega}D + L\right)x = -\left(\frac{\omega - 1}{\omega}D + U\right)x + b$$

a kada to preoblikujemo u iterativan postupak, gdje lijevu stranu rješavamo za $x^{(k+1)}$ pomoću prethodne vrijednosti $x^{(k)}$ na desnoj strani, imamo

$$D\left(\frac{1}{\omega}I + \tilde{L}\right)x^{(k+1)} = -D\left(\frac{\omega - 1}{\omega}I + \tilde{U}\right)x^{(k)} + b$$

gdje sad imamo $\tilde{L} = D^{-1}L$ kao donjetrokutastu matricu i $\tilde{U} = D^{-1}U$ kao gornjetrokutastu matricu, a I je jedinična matrica.
Kada to malo sredimo, imamo sljedeći izraz:

$$\begin{aligned} \left(I + \omega\tilde{L}\right)x^{(k+1)} &= -\left((\omega - 1)I + \omega\tilde{U}\right)x^{(k)} + D^{-1}\omega b \\ &= \left((1-\omega)I - \omega\tilde{U}\right)x^{(k)} + D^{-1}\omega b \end{aligned}$$

te na kraju imamo

$$x^{(k+1)} = \left(I + \omega\tilde{L}\right)^{-1}\left((1-\omega)I - \omega\tilde{U}\right)x^{(k)} + \left(I + \omega\tilde{L}\right)^{-1}D^{-1}\omega b, \tag{2.1}$$

pri čemu je $k \in \mathbb{N} \cup \{0\}$.

Uzimajući prednost trokutastog oblika matrice $(I + \omega\tilde{L})$, element $x^{(k+1)}$ možemo računati serijski, tj. sekvencionalno koristeći supstitucije unaprijed, te iz toga slijedi izraz

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} - \frac{\omega}{a_{ii}}\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) + \omega\frac{b_i}{a_{ii}}, \qquad i = 1, \ldots, n$$

Iz izraza (2.1) možemo iščitati formulu za matricu konvergencije (1.11) SOR metode, u oznaci $C_{SOR(\omega)}$.

$$C_{SOR(\omega)} = \left(I + \omega\tilde{L}\right)^{-1}\left((1-\omega)I - \omega\tilde{U}\right) \tag{2.2}$$

Vrlo lako možemo zaključiti da se dobra vrijednost za ω može unaprijed odrediti za sve iteracije te da vrijedi $\omega > 1$, što opravdava naziv OR u samom nazivu metode.
O uskoj povezanosti, tj. preciznije ovisnosti konvergencije metode i optimalnog izbora parametra ω, govorit ćemo puno detaljnije u sljedeća dva potpoglavlja.

2.3 Konvergencija SOR metode

Teorem 2.3.1 *Za proizvoljnu matricu A vrijedi*

$$\rho(C_{SOR(\omega)}) \geq |\omega - 1|$$

pa je $0 < \omega < 2$ nužan uvjet konvergencije SOR metode.
Ako je A pozitivno definitna, $0 < \omega < 2$ je također dovoljan uvjet konvergencije.

Dokaz. Znamo da je determinanta matrice jednaka produktu svih njezinih svojstvenih vrijednosti. Izračunajmo determinantu matrice konvergencije SOR metode (2.2) koja je produkt trokutastih matrica. Iskoristimo Binet-Cauchyjev teorem i činjenicu da su $\tilde{L}$ i $\tilde{U}$ strogo trokutaste matrice. Zbog toga, samo dijagonale, tj. članovi s I ulaze u determinante, pa je

$$\begin{aligned} \det C_{SOR(\omega)} &= \det(I + \omega\tilde{L})^{-1} \cdot \det((1-\omega)I - \omega\tilde{U}) \\ &= \det I \cdot \det((1-\omega)I) \\ &= (1-\omega)^n \end{aligned}$$

S druge strane je

$$\det C_{SOR(\omega)} = \prod_{j=1}^{n} \lambda_j(C_{SOR(\omega)}),$$

pa iz $|\lambda_j(C_{SOR(\omega)})| \leq \rho(C_{SOR(\omega)})$ dobivamo

$$|1-\omega|^n = \prod_{j=1}^{n} |\lambda_j(C_{SOR(\omega)})| \leq (\rho(C_{SOR(\omega)}))^n,$$

odakle slijedi $\rho(C_{SOR(\omega)}) \geq |\omega - 1|$. Dakle, za konvergenciju SOR(ω) metode mora vrijediti $|\omega - 1| < 1$, odnosno $0 < \omega < 2$. □

Teorem 2.3.2 (Ostrowski) *Neka je $A \in \mathbb{R}^{n\times n}$ simetrična pozitivno definitna matrica i pretpostavimo da je $0 < \omega < 2$. Tada SOR metoda konvergira prema jedinstvenom rješenju od $Ax = b$.*

Dokaz. Neka je λ proizvoljna svojstvena vrijednost matrice $C_{SOR(\omega)}$ i neka je $y \neq 0$ odgovarajući svojstveni vektor. Tada vrijedi

$$((1-\omega)I - \omega\tilde{U})y = \lambda(I + \omega\tilde{L})y. \tag{2.3}$$

Za lijevu stranu gornje jednakosti, direktnim računom dobivamo

$$2[(1-\omega)I - \omega\tilde{U}] = (2-\omega)I - \omega A - \omega(\tilde{U} - \tilde{L}).$$

A za desnu stranu vrijedi

$$2(I + \omega\tilde{L}) = (2-\omega)I + \omega A + \omega(\tilde{U} - \tilde{L})$$

Poslije skalarnog množenja sa y, zbog uvjeta teorema da je matrica A simetrična pozitivno definitna matrica, odnosno da je $\tilde{L} = \tilde{U}^*$, iz (2.3) dobivamo

$$\lambda = \frac{(2-\omega)d - \omega a + i\omega s}{(2-\omega)d + \omega a + i\omega s}$$

gdje je

$$a := (Ay, y), \quad d := (Dy, y), \quad s := i(\tilde{U}y - \tilde{L}y, y).$$

Kako je A pozitivno definitna, imamo $a > 0$ i $d > 0$ i jer je A simetrična, slijedi da je $s \in \mathbb{R}$. Iz

$$|(2-\omega)d - \omega a| < |(2-\omega)d + \omega a|$$

za $0 < \omega < 2$ sada možemo zaključiti da je $|\lambda| < 1$. Stoga konvergencija SOR metode za $0 < \omega < 2$ slijedi iz Teorema 1.2.1 □

Teorem 2.3.3 *SOR metoda ne konvergira za $\omega \leq 0$ i $\omega \geq 2$.*

Dokaz. Očito je da za matricu konvergencije SOR metode (2.2) vrijedi

$$\begin{aligned} \det C_{SOR(\omega)} &= \det[(I + \omega\tilde{L})^{-1}((1-\omega)I - \omega\tilde{U})] \\ &= (\det I)^{-1}(1-\omega)^n \det I \\ &= (1-\omega)^n, \end{aligned}$$

jer su $I + \omega\tilde{L}$ i $(1-\omega)I - \omega\tilde{U}$ donje trokutasta i gornje trokutasta matrica, a $\tilde{L}$ i $\tilde{U}$ strogo donje odnosno strogo gornje trokutasta matrica. Kako je

$$\rho(C_{SOR(\omega)})^n \geq \prod_{j=1}^{n} |\lambda_i| = |\det C_{SOR(\omega)}| = |1-\omega|^n$$

gdje su λ_j svojstvene vrijednosti matrice $C_{SOR(\omega)}$, slijedi

$$\rho(C_{SOR(\omega)}) \geq |1-\omega|$$

Prema Teoremu 1.2.1 SOR metoda će konvergirati ako i samo ako je $\rho(C_{SOR(\omega)}) < 1$, pa ako imamo da je $|1-\omega| \geq 1$, tada SOR metoda neće konvergirati. Ovaj uvjet biti će ispunjen ako i samo ako je $\omega \leq 0$ i $\omega \geq 2$. □

2.4 Izbor relaksacijskog parametra

Do sada smo se već okvirno upoznali s relaksacijskim parametrom kao realnim brojem različitim od nule, te smo dobili osjećaj zašto se javila potreba za njim. No, nismo ništa rekli o njegovom izboru, koji ovisi o svojstvima dane matrice. Ovdje ćemo se malo više pozabaviti time.

Na početku prvog poglavlja vidjeli smo da brzina konvergencije iterativne metode ovisi o spektralnom radijusu $\rho(C)$ gdje je C matrica konvergencije pojedine metode dana u (1.11). S obzirom da mi nismo zainteresirani samo za konvergenciju, nego za što bržu konvergenciju, manji $\rho(C)$ će nam ju i osigurati (Napomena 1.2.1).
S obzirom da početni vektor iteracije sami možemo birati, zaključujemo da time možemo dobiti i manju grešku pri njegovom boljem izboru, ali svakako znamo da je ocjena (1.12) dostižna. Dakle, da bismo globalno ubrzali konvergenciju metode trebamo dobiti što manji spektralni radijus $\rho(C)$. U tom smislu, kod SOR(ω) metode, one vrijednosti parametra ω za koje je $\rho(C_{SOR(\omega)})$ globalno najmanji, zovemo optimalnim relaksacijskim parametrima i označavamo s ω_{opt}, kao što smo već ranije napomenuli.
U nastavku ćemo pokazati koji je to optimalni izbor parametra koji minimizira SOR, no najprije navedimo par definicija koje će nam trebati pri razumijevanju teorema.

Teorem 2.4.1 *Matrica A reda n je reducibilna ako postoji matrica permutacije P takva da je*

$$PTP^T = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

pri čemu su A_{11} i A_{22} kvadratni blokovi reda manjeg od n, tj. PAP^T je blok gornjetrokutasta matrica. Matrica A je ireducibilna, ukoliko nije reducibilna, tj. ukoliko ne postoji matrica permutacije P za koju je PAP^T blok gornjetrokutasta matrica.

Definicija 2.4.1 *Matrica A je slabo dijagonalno dominantna po recima ako za svaki i vrijedi*

$$|a_{ii}| \geq \sum_{\substack{k=1 \\ k \neq i}}^{n} |a_{ik}|,$$

pri čemu se stroga nejednakost javlja barem jednom.

Teorem 2.4.2 *Pretpostavimo da matrica $A = I - L - U$ ima sljedeće svojstvo: za svaki $c \in \mathbb{R}$,*

$$\det(cI - L - U) = \det(cI - \gamma L - \gamma^{-1} U) \tag{2.4}$$

za svaki $\gamma \in \mathbb{R} \backslash \{0\}$. Tada vrijedi sljedeće:

(a) Ako je μ svojstvena vrijednost od C_J, tada je $-\mu$ također svojstvena vrijednost od C_J s istom kratnošću.

(b) Ako je $\lambda = 0$ svojstvena vrijednost $C_{SOR(\omega)}$, tada je $\omega = 1$.

(c) Ako je $\lambda \neq 0$ svojstvena vrijednost od $C_{SOR(\omega)}$ za neki $\omega \in \langle 0, 2 \rangle$, tada je

$$\mu = \frac{\lambda + \omega - 1}{\omega \lambda^{1/2}} \tag{2.5}$$

svojstvena vrijednost od C_J.

(d) Ako je μ svojstvena vrijednost od C_J, te λ zadovoljava (2.5) za neki $\omega \in \langle 0, 2\rangle$, tada je λ svojstvena vrijednost od $C_{SOR(\omega)}$.

Dokaz. Iz svojstva (2.4) s $\gamma = -1$, te za bilo koju vrijednost μ, imamo

$$\begin{aligned} \det(C_J - \mu I) &= \det[D^{-1}(L+U)] \\ &= \det(\tilde{L} + \tilde{U} - \mu I) \\ &= \det(-\tilde{L} - \tilde{U} - \mu I) \\ &= (-1)^n \det(\tilde{L} + \tilde{U}U + \mu I) \\ &= (-1)^n \det(C_J + \mu I). \end{aligned}$$

Kako su svojstvene vrijednosti matrice C_J brojevi μ za koje je $\det(C_J - \mu I) = 0$ i njihovi višekratnici su također određeni ovim karakterističnim polinomom, svojstvo pod a) je dokazano.
Kako je matrica $I - \omega\tilde{L}$ donje trokutasta s jedinicama na dijagonali, njena determinanta je 1, stoga za bilo koji λ imamo:

$$\begin{aligned} \det(C_{SOR(\omega)} - \lambda I) &= \det[(I - \omega\tilde{L})^{-1}[(1-\omega)I + \omega\tilde{U}] - \lambda I] \\ &= \det[(1-\omega)I + \omega\tilde{U} - \lambda(I - \omega\tilde{L})]. \end{aligned} \tag{2.6}$$

Ako je $\lambda = 0$ svojstvena vrijednost od $C_{SOR(\omega)}$, tada (2.6) implicira da $\det[(1-\omega)I + \omega\tilde{U}] = 0$. S obzirom da je ta matrica gornje trokutasta s elementima $(1-\omega)$ na dijagonali, zaključujemo da je $\omega = 1$, te je stoga dokazana tvrdnja pod b).
Za $\lambda \neq 0$, jednakost (2.6) implicira sljedeće:

$$\det(C_{SOR(\omega)} - \lambda I) = \omega^n \lambda^{n/2} \det\left[\frac{1-\omega-\lambda}{\omega\lambda^{1/2}} I + \lambda^{1/2}\tilde{U} + \lambda^{1/2}\tilde{L}\right].$$

Koristeći svojstvo (2.4) iz pretpostavke teorema s $\gamma = \lambda^{-1/2}$, imamo:

$$\det(C_{SOR(\omega)} - \lambda I) = \omega^n \lambda^{n/2} \det\left[C_J - \frac{\lambda + \omega - 1}{\omega\lambda^{1/2}} I\right].$$

Slijedi da ako je $\lambda \neq 0$ svojstvena vrijednost od $C_{SOR(\omega)}$ i μ zadovoljava jednakost (2.5), tada je μ svojstvena vrijednost od C_J. Obrnuto, ako je μ svojstvena vrijednost od C_J i λ zadovoljava (2.5), tada je λ svojstvena vrijednost od $C_{SOR(\omega)}$. Time su dokazane tvrdnje teorema pod c) i d). □

Teorem 2.4.3 *Pretpostavimo da matrica A zadovoljava svojstvo (2.4), te neka matrica C_J ima realne svojstvene vrijednosti i neka je $\beta \equiv \rho(C_J) < 1$. Tada SOR metoda konvergira za svaki $\omega \in \langle 0, 2\rangle$, a spektralni radijus matrice konvergencije SOR metode je*

$$\rho(C_{SOR(\omega)}) = \begin{cases} \frac{1}{4}\left[\omega\beta + \sqrt{(\omega\beta)^2 - 4(\omega-1)}\right]^2, & 0 < \omega \leq \omega_{opt}, \\ \omega - 1, & \omega_{opt} \leq \omega \leq 2 \end{cases} \tag{2.7}$$

gdje je ω_{opt} optimalna vrijednost od ω, te iznosi

$$\omega_{opt} = \frac{2}{1 + \sqrt{1-\beta^2}}. \tag{2.8}$$

Za svaku drugu vrijednost od ω, imamo

$$\rho(C_{SOR(\omega_{opt})}) < \rho(C_{SOR(\omega)}), \quad \omega \in \langle 0, 2\rangle \backslash \{\omega_{opt}\}.$$

Dokaz. Rješavanjem jednadžbe (2.5) po λ dobivamo

$$\lambda = \frac{1}{4}\left(\omega\mu \pm \sqrt{(\omega\mu)^2 - 4(\omega-1)}\right)^2. \tag{2.9}$$

Iz Teorema 2.4.2 slijedi da ako je μ svojstvena vrijednost matrice C_J, tada su oba korijena λ svojstvene vrijednosti od $C_{SOR(\omega)}$. S obzirom da je μ realan broj, izraz pod korijenom u (2.9) je negativan ukoliko je

$$\tilde{\omega} \equiv \frac{2(1-\sqrt{1-\mu^2})}{\mu^2} < \omega < 2,$$

a u ovom slučaju vrijedi

$$\begin{aligned} |\lambda| &= \frac{1}{4}\left[(\omega\mu)^2 + 4(\omega-1) - (\omega\mu)^2\right] \\ &= \omega - 1, \quad \omega \in \langle\tilde{\omega}, 2\rangle. \end{aligned}$$

Za preostale vrijednosti ω, odnosno za preostale dijelove intervala, oba rješenja kvadratne jednadžbe su pozitivna, a veći λ iznosi

$$\frac{1}{4}\left[(\omega|\mu|) + \sqrt{(\omega|\mu|)^2 - 4(\omega-1)}\right]^2, \quad \omega \in \langle 0, \tilde{\omega}]. \tag{2.10}$$

Također, ova je vrijednost λ veća ili jednaka $\omega - 1$ za $\omega \in \langle 0, \tilde{\omega}]$ jer u tom rasponu imamo

$$\frac{1}{4}\left[(\omega|\mu|) + \sqrt{(\omega|\mu|)^2 - 4(\omega-1)}\right]^2 \geq \frac{1}{4}(\omega|\mu|)^2 \geq \omega - 1 \tag{2.11}$$

Lako je provjeriti da za svaki fiksni $\omega \in \langle 0, \tilde{\omega}]$, izraz (2.10) je strogo rastuća funkcija po varijabli $|\mu|$. Također, $\tilde{\omega}$ je strogo rastuća funkcija po $|\mu|$, pa imamo

$$\tilde{\omega} \leq \frac{2(1-\sqrt{1-\beta^2})}{\beta^2} = \frac{2}{1+\sqrt{1-\beta^2}} \equiv \omega_{opt}.$$

Slijedi da svojstvena vrijednost λ od $C_{SOR(\omega)}$ za koju je $|\lambda| = \rho(C_{SOR(\omega)})$, odgovara svojstvenoj vrijednosti μ od C_J za koju je $|\mu| = \beta$ jer je takva svojstvena vrijednost veća ili jednaka od onih koje odgovaraju manjim vrijednostima od $|\mu|$ ako je $\omega \in \langle 0, \omega_{opt}]$, a jednaka je ostalim ako je $\omega \in \langle \omega_{opt}, 2\rangle$.
Zaključujemo da (2.7) vrijedi za ω_{opt} dan s (2.8). S obzirom da je izraz (2.8) manji od 1 za sve $\omega \in \langle 0, 2\rangle$, SOR metoda konvergira. Također se vidi da je za fiksni $|\mu| = \beta$, izraz (2.10) strogo padajuća funkcija od ω za $\omega \in \langle 0, \omega_{opt}]$ koja postiže minimum u $\omega = \omega_{opt}$.
Posljednja nejednakost teorema,

$$\rho(C_{SOR(\omega_{opt})}) < \rho(C_{SOR(\omega)}), \quad \omega \in \langle 0, 2\rangle \backslash \{\omega_{opt}\}$$

je tada dokazana. □

3 Usporedba brzine konvergencije i točnosti SOR, Jacobi i Gauss-Seidelove metode za nekoliko klasa problema u programskom jeziku MATLAB

U ovom posljednjem poglavlju bit će sumirani kriteriji koji garantiraju konvergenciju ove tri metode, te će biti primijenjeni i uspoređeni na primjerima. Svi izračuni napravljeni su u programskom jeziku MATLAB. Potpuni kod MATLAB implementacije pojedinih metoda navedenih u ovom radu bit će dan kroz sljedeća potpoglavlja, a MATLAB kod grafičkog korisničkog sučelja korištenog u sljedećim primjerima se nalazi na kraju ovog diplomskog rada, u poglavlju Dodatak.

Prije nego se upustimo u analizu konvergencije iterativnih metoda, osvrnimo se na složenost iterativnih metoda, odnosno broj aritmetičkih operacija potrebnih za izračun aproksimacije rješenja koja zadovoljava željenu toleranciju. Kako je napomenuto u prvom poglavlju, složenost direktnih metoda je reda veličine $O(n^3)$. Pogledajmo složenost iterativnih metoda. U komponentnim iterativnim metodama, Jacobijevoj, Gauss-Seidelovoj i SOR metodi, vrijede sljedeće relacije:

$$\begin{aligned}
x_i^{(k+1)} &= \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right),\\
x_i^{(k+1)} &= \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1} a_{ij}x_j^{(k)}\right),\\
x_i^{(k+1)} &= (1-\omega)x_i^{(k)} - \frac{\omega}{a_{ii}}\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) + \omega\frac{b_i}{a_{ii}}, \quad i = 1,\ldots,n.
\end{aligned}$$

Ako je A puna matrica, s reda veličine n elemenata različitih od nule u svakom retku, tada računanje svake komponente zahtijeva $O(n)$ aritmetičkih operacija, zbrajanja, oduzimanja, množenja i dijeljenja. Kada sagledamo ukupnu složenost, za svaku novu iteraciju $x^{(k+1)}$ trebamo reda veličine $O(n^2)$ operacija, što odgovara matrično-vektorskom množenju pri čemu se vektor sastoji od n komponenti, a matrica je dimenzije $n \times n$. Stoga su iterativne metode za pune matrice uistinu brže, u usporedbi s direktnim metodama, ali samo ako je broj iteracija bitno manji od n.
Međutim, ukoliko je matrica A rijetko popunjena, s konstantnim brojem elemenata koji su različiti od nule u svakom retku, tada računanje svake komponente vektora aproksimacije zahtjeva konstantan broj iteracija, tj. za svaku novu aproksimaciju $x^{(k+1)}$ trebamo reda veličine $O(n)$ operacija.

Sagledajmo sada kriterije konvergencije Jacobijeve, Gauss-Seidelove i SOR metode navedene u ovom radu. Prvi kriterij konvergencije iterativnih metoda je jednostavno provjeriti ali nije uvijek primijenjiv u konkretnim primjerima, kao što smo i dosad naveli, no drugi kriterij daje jače uvjete na matricu A ali shodno tome i više informacija o konvergenciji, dok treći kriterij govori o konvergenciji SOR metode. Pa pogledajmo:

1. Ako je A strogo dijagonalno dominantna po recima, tada konvergiraju i Jacobijeva i Gauss-Seidelova metoda, no Gauss-Seidelova brže (Teorem 1.2.3.3).

2. Ukoliko naš promatrani problem nema strogo dijagonalno dominantnu matricu, prethodni kriterij nam ne pomaže. Tada tražimo slabu dijagonalnu dominaciju (Definicija 2.4.1), ali se nameće i ireducibilnost matrice A (Definicija 2.4.1) kako bismo dokazali konvergenciju Gauss-Seidelove i Jacobijeve metode. Ako je matrica A ireducibilna i slabo dijagonalno dominantna po recima, obje metode konvergiraju, no Gauss-Seidelova brže. (Vidi [3].)

3. Gledajući SOR(ω) metodu, došli smo do zaključka da je $0 < \omega < 2$ nužan uvjet konvergencije (Teorem 2.3.1). Ako je A pozitivno definitna, $0 < \omega < 2$ je također dovoljan uvjet konvergencije.

Prije nego pogledamo primjenu navedenih kriterija na primjerima, pogledajmo implementaciju Jacobijeve, Gauss-Seidelove i SOR metode u programskom jeziku MATLAB.

3.1 Implementacija Jacobijeve, Gauss-Seidelove i SOR metode u programskom jeziku MATLAB

Kao što je već u uvodu ovog diplomskog rada rečeno, MATLAB je programski jezik visoke razine i interaktivna okolina za numeričko i matrično računanje, te za vizualizaciju i programiranje. Sam naziv je nastao kao kratica od engleskih riječi MATrix LABoratory. Upravo zbog mogućnosti manipuliranja matricama, iscrtavanja funkcija i podataka, te implementacije algoritama i stvaranje grafičkih korisničkih sučelja nametnuo se kao izvrstan alat koji je poslužio u izradi ovog diplomskog rada.
Pogledajmo prvo implementacije pojedinih metoda.

3.1.1 Implementacija Jacobijeve metode

```
function [koraci, spradius]=jacobi(A,x,b, tolerancija)

d=diag(A);
D=diag(d);
LU=A-D;
[m,n] = size(A);

for ii=1:n
    for jj=1:m
       C(ii,jj)=-(1/D(ii,ii))*LU(ii,jj);
    end
end
eig(C)
spradius=max(abs(eig(C)));

koraci=0;

while norm(A*x-b, 2) >= tolerancija
    for kk=1:n
       x1=(1/D(kk,kk))*(b-LU*x);
    end
    x=x1;
```

```
    koraci=koraci+1;
    greska(koraci)=norm(A*x-b, 2);
end

disp(sprintf('Broj iteracija Jacobijeve metode: %d', koraci))
semilogy(greska, 'o-g')
```

Jacobijeva metoda je ovdje definirana kao funkcija koja prima vrijednosti matrice sustava A, početnu aproksimaciju x, vektor b, te toleranciju. Ti parametri će biti zadani tijekom generiranja primjera u grafičkom korisničkom sučelju. U prve dvije linije koda definiramo matrice koje predstavljaju rastav matrice A. Matrica C je Jacobijeva matrica konvergencije, te provjeravamo njezin spektralni radijus kako bismo dobili uvid u konvergenciju metode za danu matricu A. Ako je njezin spektralni radijus manji od 1, tada metoda konvergira. Pri ulasku u while petlju nailazimo na njezin uvjet koji provjerava je li Euklidska norma reziduala veća ili jednaka toleranciji. Dokle god je ona veća ili jednaka toleranciji, while petlja se izvršava. Unutar nje definiramo for petlju unutar koje izračunavamo sljedeću aproksimaciju. Pri izlasku iz for petlje, vrijednost prijašnje aproksimacije prebrišemo sadašnjom aproksimacijom te korak iteracije povećamo za jedan. Nakon prolaska kroz while petlju, dobili smo aproksimaciju sa željenom toleranciju, te ispisujemo broj koraka Jacobijeve metode i crtamo grešku u ovisnosi o iteracijama.

3.1.2 Implementacija Gauss-Seidelove metode

```
function[koraci, spradius]=(A,x,b, tolerancija)

D=diag(diag(A));
LD =tril(A);
U=A-(LD);
[m,n] = size(A);
C=zeros(n);

for kk= 1:n
  for ii = 1:n
    C(kk,ii) = (-U(kk,ii) - LD(kk,1:(kk-1))*C(1:(kk-1),ii))/LD(kk,kk);
  end
end

spradius=max(abs(eig(C)));
koraci=0;

x1=x;
while norm(A*x-b, 2) >= tolerancija
  for i = 1:n
   x1(i) = (1/A(i, i))*(b(i) - A(i, 1:n)*x1 +  A(i, i)*x1(i));
  end
  x=x1;
  koraci=koraci + 1;
  greska(koraci)=norm(A*x-b, 2);
end
```

```
disp(sprintf('Broj iteracija Gauss-Seidelove metode: %d', koraci))
semilogy(greska, 'o-r')
```

Kao i kod Jacobijeve metode, prvo definiramo rastav matrice A na dijagonalnu matricu D, donjetrokutastu matricu LD i strogo gornje trokutastu matricu U. Potom računamo matricu konvergencije Gauss-Seidelove metode prolazeći kroz dvije for petlje, a nakon toga izračunavamo spektralni radijus te iste matrice. Spektralni radijus će nam reći hoće li metoda za dane vrijednosti konvergirati. While petlja se izvršava dokle god je Euklidska norma reziduala veća ili jednaka danoj toleranciji. Unutar for petlje računamo sljedeću aproksimaciju. Nakon završetka for petlje staroj aproksimaciji pridružujemo vrijednost nove aproksimacije, te povećavamo broj koraka iteracije za 1. Nakon što while petlja završi, imamo aproksimaciju do na željenu toleranciju, te ispisujemo broj koraka Gauss-Seidelove metode i crtamo grešku kroz iteracije.

3.1.3 Implementacija SOR metode

```
function [koraci, spradius]= (A,x, b, N, w, tolerancija)\\

D = diag(diag(A));
L = tril(-A,-1);
U = triu(-A,1);
[m,n]=size(A);
Cw=zeros(n);
bw=[0; 0; 0];

for ii= 1:n
   for jj = 1:n
       Cw(ii,jj) = ((1-w)*D(ii,jj)+w*U(ii,jj) - ...
                   (D(ii,1:(ii-1))-w*L(ii,1:(ii-1)))*Cw(1:(ii-1),jj)) ...
                    /(D(ii,ii)-w*L(ii,ii));
   end
end

for ii= 1:n
      bw(ii) = (w*b(ii) -  (D(ii,1:(ii-1))-w*L(ii,1:(ii-1)))*bw(1:(ii-1))) ...
               /(D(ii,ii)-w*L(ii,ii));
end

koraci=1;

while k <= N
x(:,k+1)=Cw*x(:,k) + bw;
if norm(x(:,k+1)- x(:,k)) < = tol
    'Uspjesan postupak!'
    'Broj iteracija SOR metode: '
     disp(k);
    'Aproksimacija rjesenja: '
     disp(x(:,k+1));
```

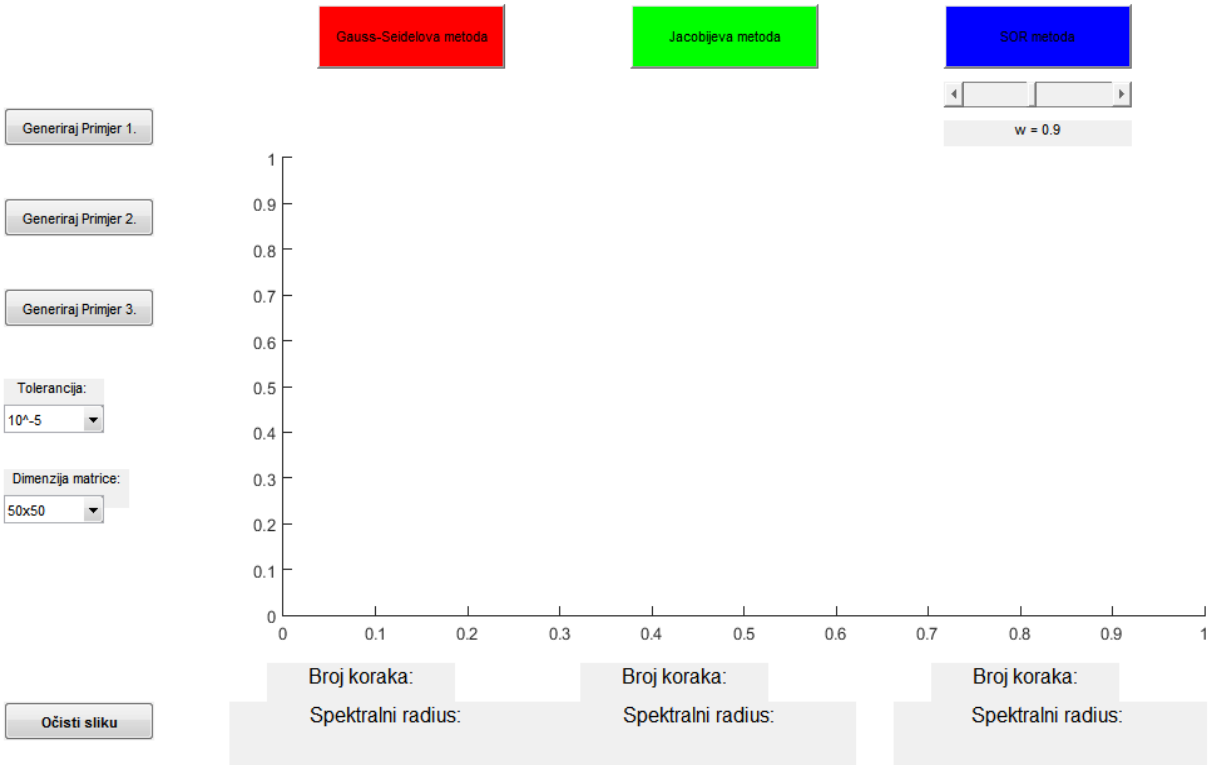
```
    break
end
koraci=koraci+1;
end
```

Ulazni podaci i u SOR funkciji su matrica A, vektor početne aproksimacije x, vektor b i tolerancija, no također i proizvoljan broj N, tj. maksimalan broj iteracija, te $\omega \in (0, 2)$. Rezultat implementacije gornjeg koda je broj koraka SOR metode, te aproksimacija rješenja nakon k koraka iteracije.
Vektor bw je konstantan vektor potreban za iteracije, a matrica Cw je matrica konvergencije SOR metode.

Preostaje još pogledati grafičko korisničko sučelje koje će biti korišteno u primjerima usporedbe brzine konvergencije iterativnih metoda navedenih u ovom radu.

3.1.4 Grafičko korisničko sučelje za iterativne metode

Grafičko korisničko sučelje, odnosno kraće GUI, napravljeno je kako bi korisniku olakšali pregled konvergencije pojedinih metoda.



Slika 1: GUI za grafički prikaz ovisnosti greške o iteracijama

Odabirom "Generiraj Primjer 1.", generira se primjer sa zadanom matricom A, vektorom b, početnom aproksimacijom x_0, te vrijednosti parametra rekalsacije SOR metode i tolerancije. Isto vrijedi i za odabir "Generiraj Primjer 2.", te "Generiraj Primjer 3.".
Odabirom padajućeg izbornika "Tolerancija", može se birati vrijednost tolerancije koja se

potom primjenjuje u izvršavanju pojedinih metoda.
Odabirom dimenzije matrice, mijenjamo dimenziju matrice sustava. Odabrati možemo dimenzije 50×50, 100×100, 200×200, te 500×500. U središtu ove slike je koordinatni sustav, logaritamski skaliran, unutar kojega se, odabirom pojedine metode, prikazuje ovisnost pogreške o broju iteracija za tu odabranu metodu. Uz SOR metodu postoji još i engl. slider, odnosno klizna traka. Pomićući kliznu traku, mijenjamo vrijednost ω relaksacijskog parametra SOR metode, te se njegova vrijednost ispisuje ispod klizne trake. Na dnu ovog prikaza nalaze se ispisi za broj koraka, te spektralni radijus pojedine metode.
U lijevom donjem uglu nalazi se gumb za brisanje grafa u koordinatnom sustavu.

Pogledajmo sada kod za već spomenuti GUI u MATLAB-u. S obzirom na veličinu koda, bit će prikazani samo pojedini dijelovi, kako bi što lakše bilo za shvatiti njegovu implementaciju. Cijeli kod se nalazi u dodatku ovom diplomskom radu.
Na samom početku, prvo definiramo poziciju i veličinu okvira korisničkog sučelja, a potom i koordinatnih osi unutar GUI-a.

```
f=figure('position', [150 60 1050 610],...
           'name', 'Iterativne metode');
set(gca,'position', [0.23 0.19 0.7 0.58]);
P=0;
```

Prva komponenta unutar uglatih zagrada govori o poziciji sučelja na x-osi, a druga na y-osi. Treća komponenta predstavlja skaliranje po x-osi, a četvrta po y-osi.
Zatim slijedi definiranje gumba. Klikom na gumb izvršavaju se metode. Varijabla 'P' koja je postavljena na vrijednost 0 predstavlja zastavicu čija će uloga biti kasnije objašnjena.

```
SOR=uicontrol('style', 'pushbutton',...
   'position', [770 540 150 50],...
   'string', 'SOR metoda',...
   'backgroundcolor', 'blue',...
   'callback',...
  ['if P==1, [k,s3]=sor(A,x,b,N,w, tol); hold on,',...
   'set(koraci3, ''string'',sprintf(''Broj iteracija: %d'',k));',...
   'set(spradius3, ''string'',sprintf(''Spektralni radijus: %.4f'',s3));',...
   'else title(''Prvo generirajte primjer!''),end']);
```

Ovo je gumb SOR metode. Naredba 'uicontrol' kreira kontrolni objekt korisničkog sučelja. Metode su definirane kao 'pushbutton', odnosno kao gumb. Svojstvom 'backgroundcolor' mijenjamo boju pozadine gumba. Najvažniji dio je svakako naredba 'callback' koja sadrži kod koji se izvršava kao odgovor na odabir gumba. Tu, ukoliko je uvjet 'P=1' ispunjen, pozivamo metodu koja vraća broj koraka, te spektralni radijus matrice iteracija pojedine metode. Potom postavljamo objekt koji ispisuje broj iteracija te spektralni radijus. Varijabla 'P' je na početku postavljena na vrijednost 0, međutim, generiranjem primjera čiji će kod biti kasnije naveden, postavljamo ju na vrijednost 1 i time zapravo u if petlji dobivamo da ukoliko primjer nije generiran, ispisuje se poruka "Prvo generirajte primjer".
Jacobijevu i Gauss-Seidelovu metodu definiramo na isti način.
Kako je bilo pokazano kroz ovaj rad, za SOR metodu vrlo je bitan odabir relaksacijskog parametra. Objekt koji prikazuje ovisnosti broja iteracija SOR metode i relaksacijskog parametra ω je 'slajder', definiran kao klizna traka. Pomicanjem klizne trake, mijenja se vrijednost relaksacijskog parametra, te se unutar koordinatnih osi, za svaku odabranu vrijednost relaksacijskog parametra, crta ovisnost greške o broju iteracija.

```
slajder=uicontrol('style', 'slider',...
        'position', [770 510 150 20],...
        'min',0, 'max', 2,...
        'value', 0,...
        'sliderstep', [0.0 0.05],...
        'callback',...
       ['svalue=get(slajder,''value''); [k, r]=sor(A,x,b,N,svalue, tol);',...
        'set(vslajdera, ''string'', [''w = '',num2str(svalue)]);',...
        'set(koraci3, ''string'',sprintf(''Broj iteracija: %d'',k));']);


vslajder=uicontrol('style', 'text',...
         'position', [770 480 150 20]);
```

Objekt 'vslajder' predstavlja tekst unutar kojega će se ispisivati vrijednost relaksacijskog parametra. Pomicanjem klizne trake, ažurira se i vrijednost objekta 'vslajder'.
Pogledajmo kako je definiran izbornik 'Tolerancija'.

```
tolerancijatekst=uicontrol('style', 'text',...
                'position', [20 210 80 30],...
                'string', 'Tolerancija: ')

tolerancija=uicontrol('style', 'popup',...
           'position', [20 190 80 30],...
           'string', {'10^(-3)', '10^(-5)', '10^(-7)', '10^(-9)'},...
           'value', 2,...
           'callback',...
           't=get(tolerancija, ''value''); tol=(1/10)^(2*t+1)')
```

Kontrolni objekt 'tolerancijatekst' ima oblik teksta, te kao takav, nema funkciju 'callback', već služi samo za objašnjenje korisniku što odabire ukoliko označi neku od vrijednosti iz izbornika ispod. Mogućnosti su navedene kao vrijednosti svojstva 'string', unutar vitičastih zagrada. Početna vrijednost je postavljena na 2, odnosno vrijednost ovog objekta koja će pisati u izborniku pokretanjem ovog GUI-a će biti 10^{-5}.
Mijenjanje dimenzije matrice se provodi na vrlo sličan način. Za kraj, pogledajmo još implementaciju jednog od primjera.

```
primjer3=uicontrol('style', 'pushbutton',...
        'position', [20 350 120 30],...
        'string', 'Generiraj Primjer 3.',...
        'callback',...
       ['cla, P=1; n=50; A= rand(n,n) +eye(n)*500;',...
        'b=rand(n,1); x = zeros(n,1); N=100000; w=0.8;',...
        'set(vslajdera, ''string'', [''w = '',num2str(w)]);',...
        'set(slajder, ''value'', w);']);
```

Objekt 'primjer3' je gumb čijim se pozivom ranije spomenuta zastavica stavlja na vrijednost 1, te se pri pozivu pojedine metode izvršava prvi dio if naredbe unutar koje se crta ovisnost greške o broju iteracija, te izračunava broj iteracija i spektralni radius. Također se definira i matrica $A1$ sa slučajno odabranim elementima iz uniformne distribucije na intervalu $\langle 0, 1\rangle$.

Matrica sustava definirana je kao produkt matrice $A1$ i njene transponirane matrice u sumi s matricom koja je produkt jedinične matrice i skalara 500 (matrica koja na dijagonali ima brojeve 500 a svi ostali elementi su joj jednaki nula) kako bi se osigurala simetričnost i stroga dijagonalna dominantnost matrice sustava. Potom se definira vektor b, vektor početne aproksimacije $x^(0)$, maksimalan broj iteracija SOR metode, vrijednost relaksacijskog parametra, te se postavlja tekst objekta 'vslajdera' za ispis vrijednosti relaksacijskog parametra.

U sljedećem potpoglavlju ćemo pogledati primjere usporedbe brzine konvergencije iterativnih metoda unutar kojih ćemo koristiti navedeno grafičko korisničko sučelje.

3.2 Usporedba brzine konvergencije Jacobijeve, Gauss-Seidelove i SOR metode na nekoliko klasa problema

Primjer 3.1 *Problem koji ćemo proučiti je slabo dijagonalno dominantna matrica koja je ujedno i ireducibilna, ali nije strogo dijagonalno dominantna.*

$$\mathbf{A} = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & 3 \\ 2 & 1 & 4 \end{bmatrix}$$

Vrlo lako možemo pokazati da je

$$\rho(C_J) = 0.7251 \quad \rho(C_{GS}) = 0.3062$$

iz čega jasno vidimo da obje metode konvergiraju, ali Gauss-Seidelova konvergira brže.

Proučimo malo ovisnost konvergencije SOR metode, Jacobijeve i Gauss-Seidelove metode. Iz Teorema 2.3.1 dobili smo da za optimalni parametar vrijedi $\omega_{opt} > 1$, no možemo primjetiti da što je $\mu = \rho(C_J)$ bliže 1, tj. što sporije konvergira Jacobijeva metoda, to je ω_{opt} bliže 2, tj. SOR metoda tada "produljuje" korak. I obratno, ako je μ blizu 0, te je Jacobijeva metoda brza, onda je ω_{opt} blizu 1, pa je optimalni SOR blizak Gauss-Seidelovoj metodi.

Primjer 3.2 *Pogledajmo sljedeću matricu*

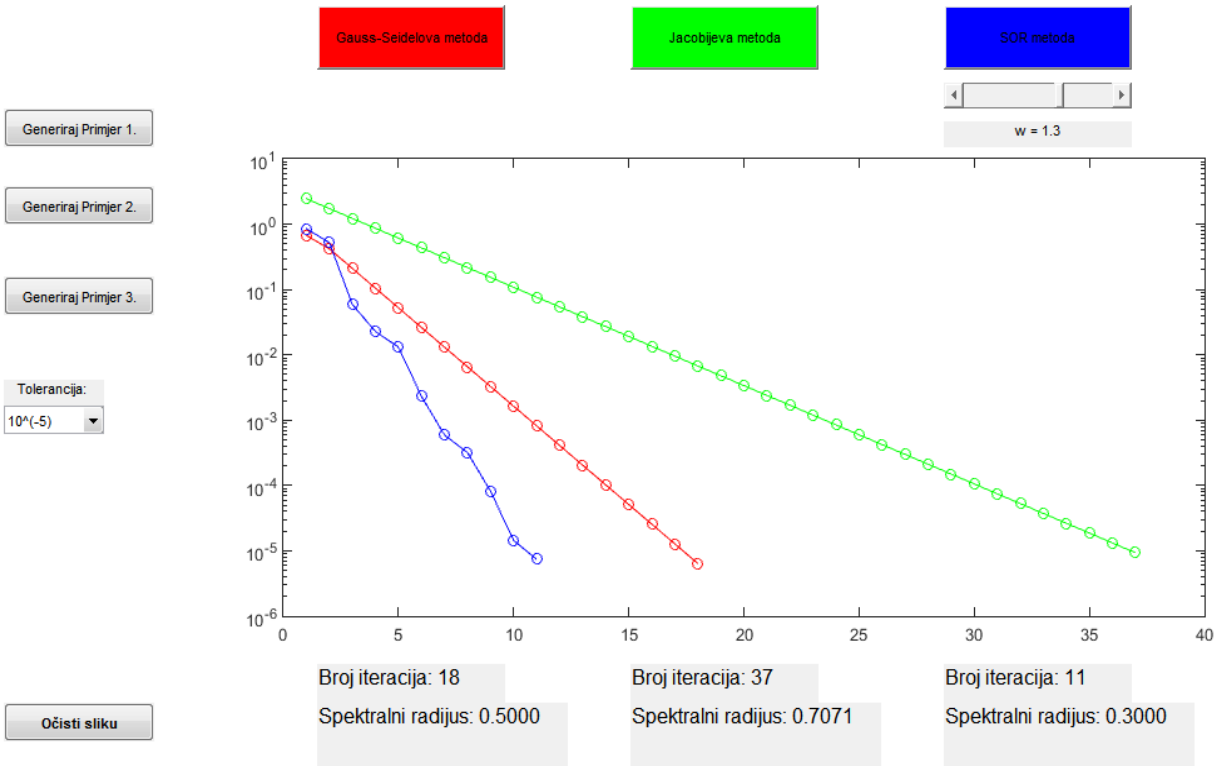
$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Matrica A je simetrična pozitivno definitna pa su stoga sve njezine svojstvene vrijednosti pozitivni realni brojevi ($\lambda_1 = 3.4142$, $\lambda_2 = 2$, $\lambda_3 = 0.5858$). Za vektor b iz sustava $Ax = b$ uzmimo $b = [-1 \quad 0 \quad -1]^T$ za početni vektor $x^{(0)}$ uzmimo nul-vektor, za ω uzmimo 1.3, te neka željena norma reziduala bude manja od 0.00001.
Nakon implementacije u MATLAB-u, dobijemo da iterativni postupak konvergira i za Gauss-Seidelovu i za SOR metodu i za Jacobijevu metodu.

$$\rho(C_{GS}) = 0.5 \qquad \rho(C_{SOR}) = 0.3 \qquad \rho(C_J) = 0.7071$$

No, možemo zaključiti da SOR metoda najbrže konvergira od preostale dvije metode.

Pogledajmo još grafički prikaz ovisnost greške o broju iteracija za navedene iterativne metode u prethodnom primjeru na Slici 2.
Greške kroz iteracije Gauss-Seidelove metode prikazane su crvenim kružićima, Jacobijeve metode zelenim, a SOR metode plavim.
Rezultat koji nam daje osjećaj za brzinu konvergencije pojedine metode je broj koraka potrebnih za smanjenje norme reziduala do zadane tolerancije, stoga se osvrnimo još i na taj rezultat. Broj koraka potreban za postizanje aproksimacije rješenja pri čemu je norma reziduala manja od 0.00001, kao što smo zadali, a računamo ju u Euklidskoj normi, pomoću Gauss-Seidelove metode iznosi 18, broj koraka za SOR metodu je 11, a broj koraka pri rješavanju sustava pomoću Jacobijeve metode je čak 37.
Ovim primjerom smo zapravo samo potvrdili tvrdnje navedene Teoremom 2.3.1. No u njemu smo, osim konvergencije Jacobijeve metode, za pretpostavku imali i da su sve svojstvene vrijednosti matrice C_J realne. Kada bismo znali da je C_J simetrična (ili hermitska), onda bi ta pretpostavka bila ispunjena, no to nije uvijek slučaj.

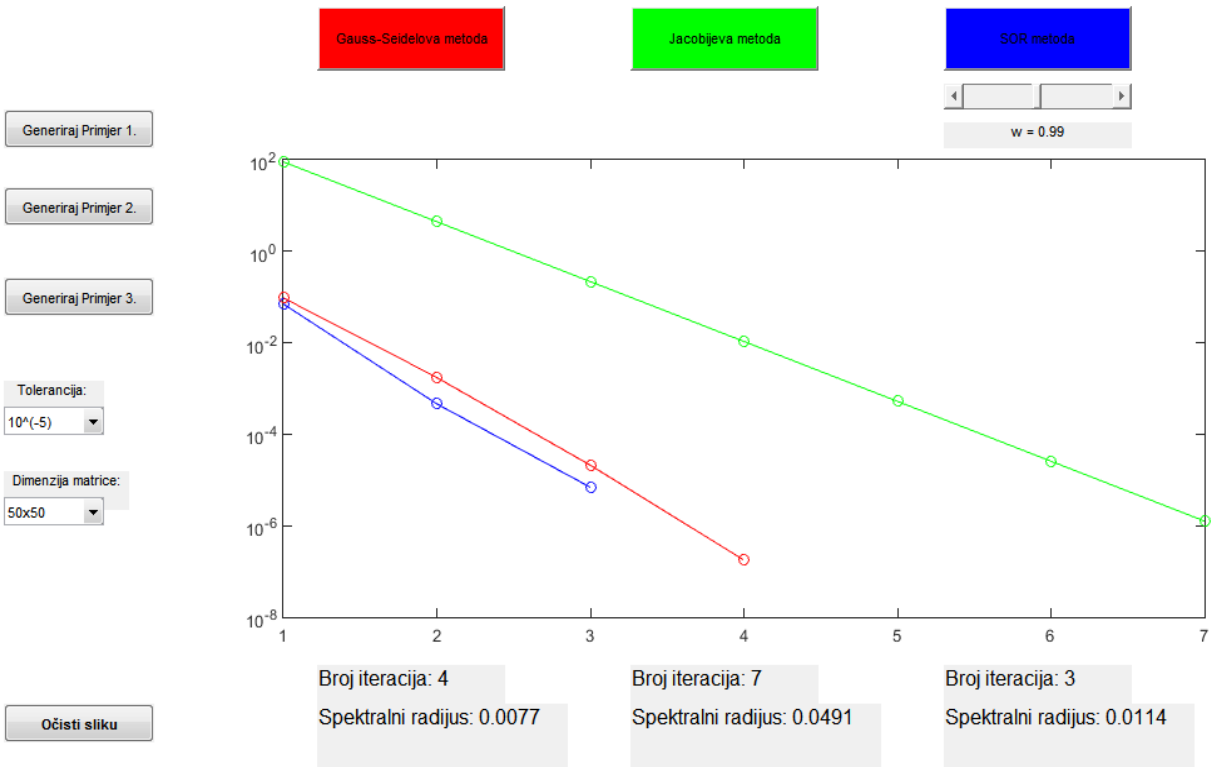


Slika 2: Ovisnost greške o broju iteracija u Primjeru 3.2

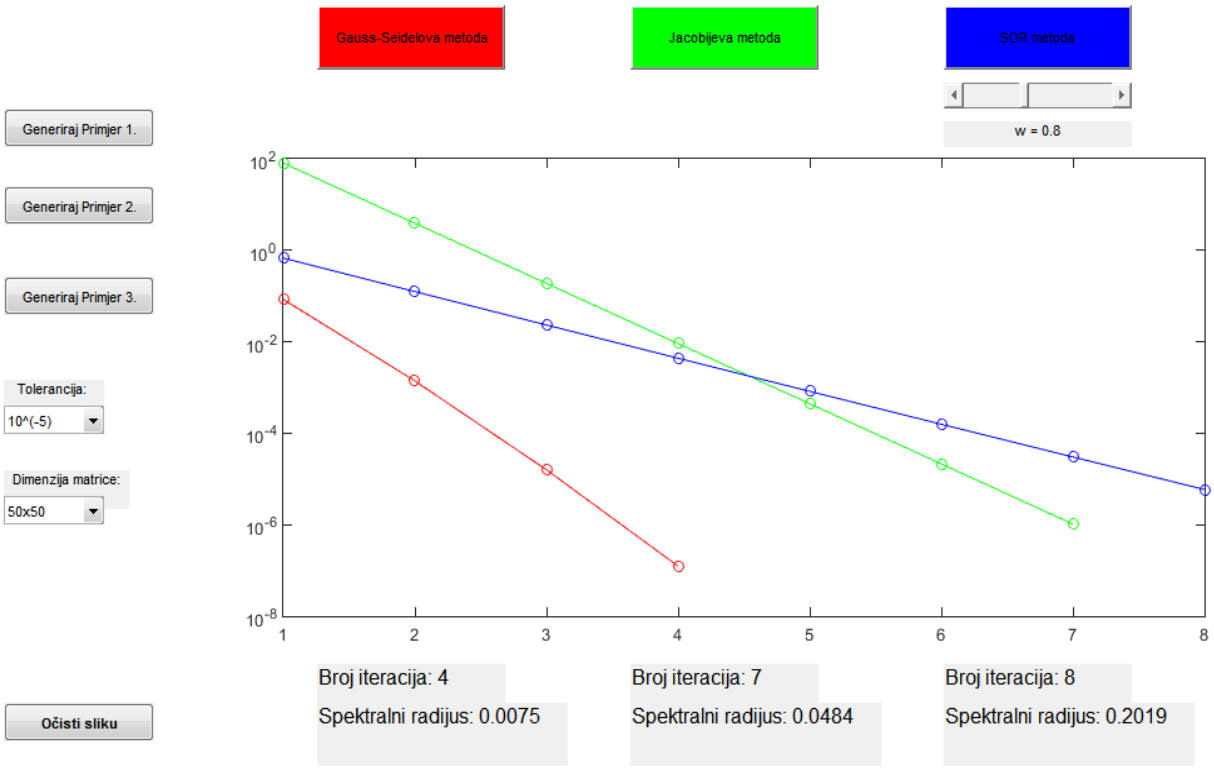
Na početku, još u uvodu, smo rekli kako se iterativne metode često koriste za rješavanje velikih rijetko popunjenih sustava, pa pogledajmo onda još dva primjera, ali ovaj put s matricom puno većih dimenzija.

Primjer 3.3 *Neka je zadana matrica 50×50 sa slučajno odabranim elementima, ali pod uvjetom da je strogo dijagonalno dominantna. Za vektor b uzmimo također slučajno odabran vektor, početna aproksimacija rješenja neka je vektor x_0 čije su komponente sve nule, a tolerancija neka bude 10^{-5} u u Euklidskoj normi. Za relaksacijski parametar uzmimo 0.99.*

Nakon implementacije ovog primjera u MATLAB-u, na Slici 3, možemo primjetiti da je broj koraka SOR metode neznatno manji od broja koraka Gauss-Sidelove metode, dok je broj koraka Jacobijeve metode je dvostruko veći u odnosu na SOR metodu.
Konvergenciju Jacobijeve metode postigli smo zahtjevajući strogu dijagonalnu dominantnost matrice sustava.
Ukoliko promijenimo vrijednost relaksacijskog parametra ω sa vrijednosti 0.99 na vrijednost 0.8, broj koraka SOR metode se uvelike povećao u odnosu na broj koraka kada je parametar imao vrijednost 0.99, kao što vidimo na Slici 4.



Slika 3: Ovisnost greške o broju iteracija u Primjeru 3.3

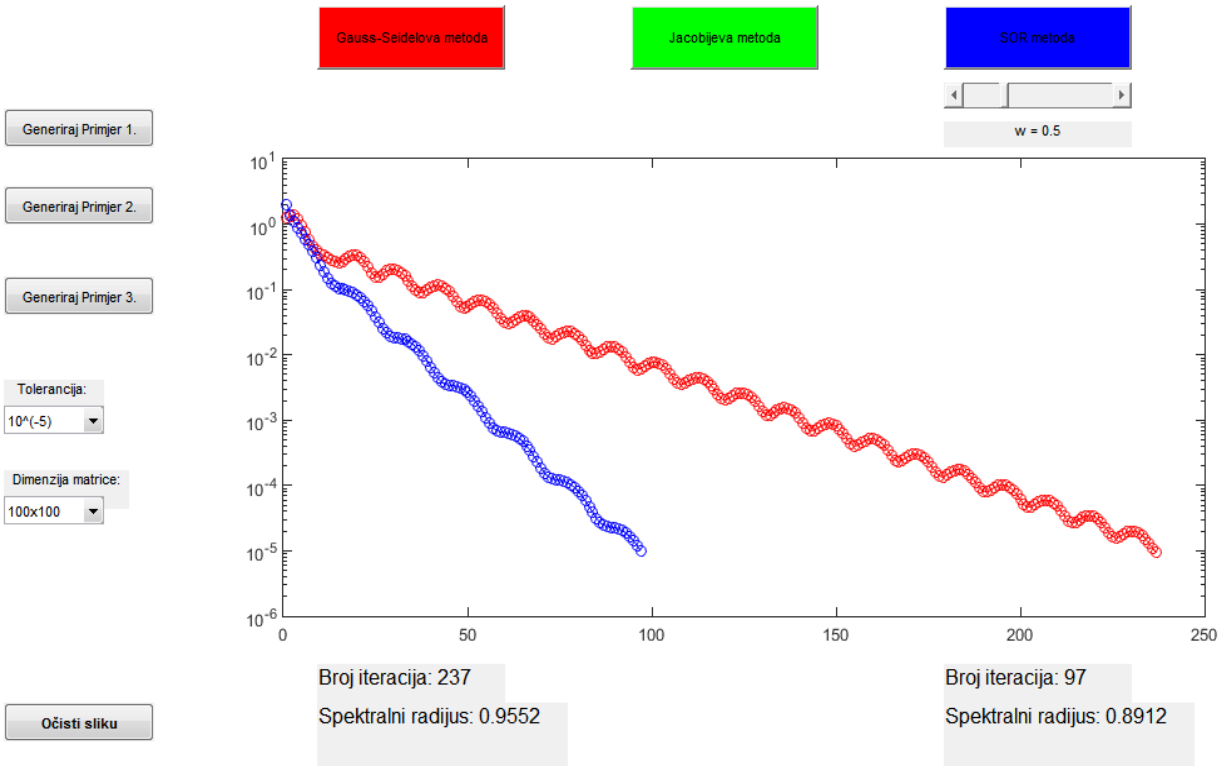


Slika 4: Ovisnost greške o broju iteracija u Primjeru 3.3

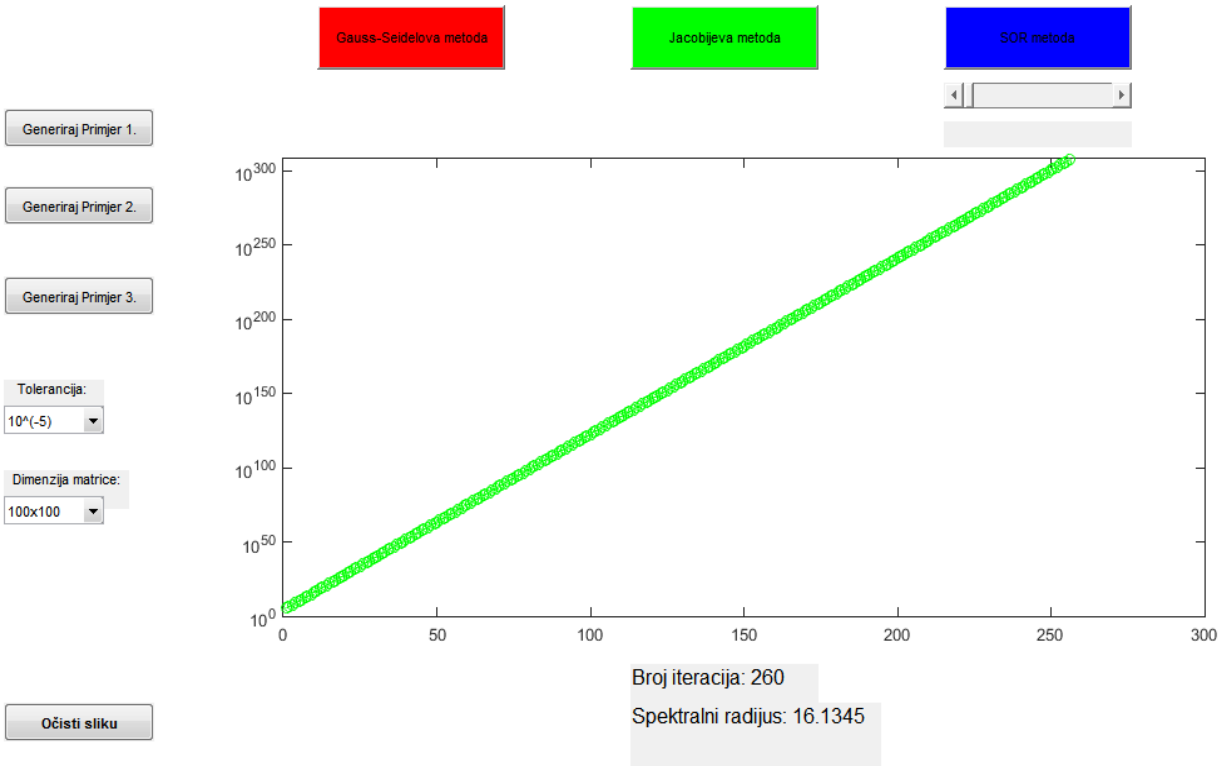
Na ovakvom primjeru, gdje je potrebno vrlo malo koraka za konvergenciju metode, teško je dobiti osjećaj koliko je to zapravo važno, te kolika je razlika u trajanju određenog iterativnog postupka. Međutim, implementacijom sljedećeg primjera pokazat ćemo da ta razlika u broju iteracija Gauss-Seidelove i SOR metode može biti dosta veća.

Primjer 3.4 *Neka je zadana matrica* 100×100 *sa slučajno odabranim elementima, ali pod uvjetom da je simetrična. Za vektor* b *uzmimo slučajno odabran vektor, početna aproksimacija rješenja neka je vektor* $x^{(0)}$ *čije su komponente sve nule, a tolerancija neka bude* 10^{-5} *u Euklidskoj normi, te neka je* $\omega = 0.5$

Primijetimo da je broj iteracija SOR metode znatno manji od broja iteracija Gauss-Sidelove metode, dok Jacobijeva metoda ne konvergira (Slika 6).
Dakle, SOR metoda je uistinu poboljšanje Gauss-Seidelove u smislu vremenske složenosti.



Slika 5: Ovisnost greške o broju iteracija u Primjeru 3.4



Slika 6: Ovisnost greške o broju iteracija u Primjeru 3.4

Kao zaključak, kroz sljedeći primjer osvrnut ćemo se na prosječan broj iteracija SOR metode na klasi simetričnih dijagonalno dominantnih matrica.

Primjer 3.5 *Neka je dano* 50 *simetričnih dijagonalno dominantnih matrica dimenzije* 100×100 *sa slučajno odabranim elementima iz uniformne distribucije na intervalu* $\langle 0, 1\rangle$*. Prosječan broj iteracija SOR metode za* $\omega = 0.8$ *iznosi 14, dok za* $\omega = 0.9$ *iznosi 15. Testiramo li program na* 100 *simetričnih dijagonalno dominantnih matrica dimenzija od* 100×100 *do* 200×200*, prosječan broj iteracija se povećava na* 30.5446 *ukoliko je relaksacijski parametar* 0.5*, a za relaksacijski* $\omega = 0.8$ *prosječan broj koraka iznosi* 40.2376.

Literatura

[1] G. Allaire, S. M. Kaber, *Numerical Linear Algebra*, Springer, 2002.

[2] N. Bosner, *Iterativne metode za rješavanje linearnih sustava*, diplomski rad, Matematički odjel, PMF, Zagreb, 2001.

[3] J. W. Demmel, *Applied Numerical Algebra*, SIAM, 1997.

[4] W. Ford, *Numerical Linear Algebra with Applications: Using MATLAB*, Academic Press, 2015.

[5] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, 1997.

[6] M. Rogina, Sanja Singer, Saša Singer, *Numerička analiza*, elektronički udžbenik

[7] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd Edition, SIAM, 2003.

[8] N. Truhar, *Numerička linearna algebra*, Odjel za matematiku, Osijek, 2010.

[9] D. M. Young, *Iterative Solution of Large Linear Systems*, Dover Publications, INC., 1971.

Sažetak

Postoje dva pristupa rješavanja sustava linearnih jednadžbi $Ax = b$ gdje za zadanu matricu $A \in \mathbb{R}^{m\times n}$ i vektor $b \in \mathbb{R}^m$ treba odrediti vektor $x \in \mathbb{R}^n$. U prvi spadaju takozvane direktne metode kod kojih se matrica A prikazuje kao produkt jednostavnijih matrica, najćešće trokutastih. To omogućava rješavanje jednostavnijih linearnih sustava, te dolazak do egzaktnog rješenja, pri čemu je složenost, odnosno broj potrebnih operacija $O(n^3)$.
U drugi pristup spadaju iterativne metode, kod kojih je pogreška $||x^{(k)} - x||$ dovoljno mala nakon znatno manje operacija od $O(n^3)$. One ne određuju rješenje direktno nego iterativno (postupno), a koriste se pri rješavanju velikih, rijetko popunjenih sustava linearnih jednadžbi, koji imaju puno nula. Takvim se metodama konstruira niz $x^{(k)}$, $k \in \mathbb{N}$ za koji vrijedi

$$x^{(k)} \to x \text{ za } k \to \infty.$$

Osnovne iterativne metode za rješavanje linearnih sustava jednadžbi su Jacobijeva metoda, te Gauss-Seidelova metoda. Metoda sukcesivne nadrelaksacije, kraće SOR metoda, je zapravo proširenje ili poboljšanje Gauss-Seidelove metode u smislu da se pomoćna nova aproksimacija $x^{(k+1)}$ računa po Gauss-Seidelovoj metodi, a sve to sa svrhom poboljšanja konvergencije. Cilj modifikacije nije proširiti klasu matrica za koje iterativna metoda konvergira već ubrzati konvergenciju, što je i pokazano na primjerima.

Ključne riječi: iterativne metode, SOR metoda, Gauss-Seidelova metoda, Jacobijeva metoda, linearni sustavi

Title and summary

Successive Overrelaxation Method (SOR)

There are two approaches to solve systems of linear equations $Ax = b$ where for the matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$ we have to determine vector $x \in \mathbb{R}^n$. The first are so-called direct method in which the matrix A is presented as a product of simpler matrices, usually triangular. It allows us to solve simpler linear systems, and to come to the exact solution, where the complexity, or the number of necessary operations is $O(n^3)$.
The second approach includes iterative methods, in which the error $||x^{(k)} - x||$ is small enough after much less operations then $O(n^3)$. Such methods do not define the solution directly, but iteratively, gradually, and we use them for solving big sparse systems of linear equations, who have a lot of zeros. With method like that, we construct sequence $x^{(k)}$, $k \in \mathbb{N}$ to which it applies

$$x^{(k)} \to x \text{ za } k \to \infty$$

The basic iterative methods for solving linear equation systems are Jacobi's method, and Gauss-Seidel's method. Successive over - relaxation method, shorter SOR method, is actually an extension or improvement of *Gauss-Seidel* method in that the auxiliary new approximation $x^{*(k+1)}$ is calculated by *Gauss-Seidel* method, and all that with purpose for improving the convergence. The goal of the modification was not to extend the class matrix for the iterative method converges, but speed up the convergence, as it is shown in examples.

Key words: iterative method, SOR method, Gauss-Seidel method, Jacobi method, convergence of methods

Životopis

Rođena sam u Vinkovcima, 06. rujna 1991. godine. Pohađala sam Osnovnu školu "Antun Gustav Matoš" Vinkovci u Vinkovcima. Nakon završene osnovne škole, upisala sam prirodoslovno-matematički smijer Gimnazije Matije Antuna Reljkovića Vinkovci u Vinkovcima, te maturirala s izvrsnim uspjehom. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja davala sam instrukcije iz matematike i sudjelovala na natjecanjima iz matematike, te se tako razvila želja za nastavkom obrazovanja u prirodoslovnom smjeru. Godine 2010. sam upisala Sveučilišni preddiplomski studij matematike na Odjelu za matematiku Sveučilišta Josipa Jurja Strossmayera u Osijeku. Budući da je nastavnički poziv samo jačao, nakon treće godine studija sam se preusmjerila na Nastavnički studij matematike i informatike.

Dodatak

U dodatku ovom dimplomskom radu nalazi se potpuni MATLAB kod za grafičko korisničko sučelje korišteno u primjerima usporedbe konvergencije Jacobijeve, Gauss-Seidelove i SOR metode.

```
clc
clear all
close all
P=0;
tol=1e-5;
n=50;
fig=figure('position', [150 60 1050 610],...
'name', 'Iterativne metode');

set(gca,'position', [0.23 0.19 0.7 0.58]);


Gauss=uicontrol('style', 'pushbutton',...
     'position', [270 540 150 50],...
     'string', 'Gauss-Seidelova metoda',...
     'backgroundcolor', 'red',...
     'callback',...
    ['if P==1, [k1, s1]=gaussseidel(A, x, b,tol); hold on,',...
     'set(koraci1,''string'',sprintf(''Broj iteracija: %d'',k1));',...
     'set(spradius1,''string'',sprintf(''Spektralni radijus: %.4f'',s1));',...
     'else title(''Prvo generirajte primjer!''),end']);

Jacobi=uicontrol('style', 'pushbutton',...
      'position', [520 540 150 50],...
      'string', 'Jacobijeva metoda',...
      'backgroundcolor', 'green',...
      'callback',...
     ['if P==1, [k2, s2]=jacobi(A,b,x, tol); hold on,',...
      'set(koraci2,''string'',sprintf(''Broj iteracija: %d'', k2));',...
      'set(spradius2,''string'',sprintf(''Spektralni radijus: %.4f'',s2));',...
      'else title(''Prvo generirajte primjer!''),end']);

SOR=uicontrol('style', 'pushbutton',...
   'position', [770 540 150 50],...
   'string', 'SOR metoda',...
   'backgroundcolor', 'blue',...
   'callback',...
  ['if P==1, [k,s3]=sor(A,x,b,N,w, tol); hold on,',...
   'set(koraci3, ''string'',sprintf(''Broj iteracija: %d'',k));',...
   'set(spradius3, ''string'',sprintf(''Spektralni radijus: %.4f'',s3));',...
   'else title(''Prvo generirajte primjer!''),end']);

koraci1=uicontrol('style', 'text',...
```

```
       'position', [270 30 150 50],...
       'horizontalalignment','left',...
       'fontsize', 12,...
       'string', 'Broj iteracija:');

koraci2=uicontrol('style', 'text',...
       'position', [520 30 150 50],...
       'horizontalalignment','left',...
       'fontsize', 12,...
       'string', 'Broj iteracija:');

koraci3=uicontrol('style','text',...
       'position', [770 30 150 50],...
       'horizontalalignment','left',...
       'fontsize', 12,...
       'string', 'Broj iteracija:' );

spradius1=uicontrol('style', 'text',...
         'position',[270 0 200 50],...
         'horizontalalignment','left',...
         'fontsize', 12,...
         'string', 'Spektralni radius:');

spradius2=uicontrol('style', 'text',...
         'position',[520 0 200 50], ...
         'horizontalalignment','left',...
         'fontsize', 12,...
         'string', 'Spektralni radius:');

spradius3=uicontrol('style','text',...
         'position', [770 0 200 50],...
         'horizontalalignment','left',...
         'fontsize', 12,...
         'string', 'Spektralni radius: ');

slajder=uicontrol('style', 'slider',...
       'position', [770 510 150 20],...
       'min',0, 'max', 2,...
       'value', 0,...
       'sliderstep', [0.05 0.05],...
       'callback',...
      ['svalue=get(slajder,''value''); [k, r]=sor(A,x,b,N,svalue, tol);',...
       'set(vslajdera, ''string'', [''w = '',num2str(svalue)]);',...
       'set(koraci3, ''string'',sprintf(''Broj iteracija: %d'',k));']);


vslajdera=uicontrol('style', 'text',...
         'position', [770 480 150 20]);
```

```
primjer1=uicontrol('style', 'pushbutton',...
        'position', [20 480 120 30],...
        'string', 'Generiraj Primjer 1.', ...
        'callback',...
       ['cla, P=1; A=[2, 1, 0; 1, 2, 1; 0, 1, 2];',...
        'b=[-1;0;-1]; x=[0;0;0]; N=1000000; w=1.3;',...
        'set(vslajdera, ''string'', [''w = '',num2str(w)]);',...
        'set(slajder, ''value'', w);']);

primjer2=uicontrol('style', 'pushbutton',...
        'position', [20 420 120 30],...
        'string', 'Generiraj Primjer 2.',...
        'callback',...
       ['cla, P=1; A1=randi([0 1],n,n); A=A1*transpose(A1) + 100*eye(n);',...
        'b=rand(n,1); x = zeros(n,1); N=100000; w=0.5;',...
        'set(vslajdera, ''string'', [''w = '',num2str(w)]);',...
        'set(slajder, ''value'', w);']);

primjer3=uicontrol('style', 'pushbutton',...
        'position', [20 350 120 30],...
        'string', 'Generiraj Primjer 3.',...
        'callback',...
       ['cla, P=1; n=50; A= rand(n,n) +eye(n)*500;',...
        'b=rand(n,1); x = zeros(n,1); N=100000; w=0.9;',...
        'set(vslajdera, ''string'', [''w = '',num2str(w)]);',...
        'set(slajder, ''value'', w);']);

ocisti=uicontrol('style', 'pushbutton',...
      'position', [20 20 120 30],...
      'string', 'Očisti sliku',...
      'fontweight', 'bold',...
      'value', 0,...
      'callback',...
     ['cla,clc,set(koraci1, ''string'', ''Broj iteracija:'');',...
      'set(koraci2, ''string'', ''Broj iteracija:'');'...
      'set(koraci3, ''string'', ''Broj iteracija:'');'...
      'set(spradius1, ''string'', ''Spektralni radius:'');',...
      'set(spradius2, ''string'', ''Spektralni radius:'');',...
      'set(spradius3, ''string'', ''Spektralni radius:'')']);

tolerancija=uicontrol('style', 'text',...
           'position', [20 270 80 30],...
           'string', 'Tolerancija: ');

tol1=uicontrol('style', 'popup',...
    'position', [20 250 80 30],...
```

```
    'string', {'10^(-3)', '10^(-5)', '10^(-7)', '10^(-9)'},...
    'value', 2,...
    'callback',...
    't=get(tol1, ''value''); tol=(1/10)^(2*t+1);');

matrica=uicontrol('style', 'text',...
       'position', [20 200 100 30],...
       'string', 'Dimenzija matrice:');

mat1=uicontrol('style', 'popup',...
    'position', [20 180 80 30],...
    'string', {'50x50', '100x100', '200x200', '500x500'},...
    'value', 1,...
    'callback',...
   ['m=get(mat1,''value''); if m==1, n=50, end;',...
    'if m==2, n=100, end;',...
    'if m==3, n=200, end;',...
    'if m==4, n=500, end;']);
```